MULTISCALE METHODS FOR ACCELERATING EXPLICIT DYNAMICS COMPUTATIONS IN SOLID MECHANICS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Gabriel José de Frías García

May 2013

MULTISCALE METHODS FOR ACCELERATING EXPLICIT DYNAMICS COMPUTATIONS IN SOLID MECHANICS

Gabriel José de Frías García, Ph.D.

Cornell University 2013

In this work we tackle two novel approaches for the solution of multiscale solid mechanics problems. In the first one a selective mass scaling approach is presented that can significantly reduce the computational cost in explicit dynamic simulations, while maintaining accuracy. One of the main computational issues with traditional explicit dynamics simulations is the significant reduction of the critical time step as the spatial resolution of the finite element mesh increases. The proposed method is based on a multiscale decomposition approach that separates the dynamics of the system into low (coarse scales) and high frequencies (fine scales). Here, the critical time step is increased by selectively applying mass scaling on the fine scale component only. In problems where the response is dominated by the coarse (low frequency) scales, significant increases in the stable time step can be realized. In this work, we use the Proper Orthogonal Decomposition (POD) method to build the coarse scale space. The main idea behind POD is to obtain an optimal low-dimensional orthogonal basis for representing an ensemble of high-dimensional data. In our proposed method, the POD space is generated with snapshots of the solution obtained from early times of the full-scale simulation. The example problems addressed in this work show significant improvements in computational time, without heavily compromising the accuracy of the results. The second approach uses POD in a similar manner, but adopts an equation-free central difference projective integration

scheme to observe and advance dynamics of the coarse scales. This equation-free approach is adopted in order to circumvent some of the drawbacks of the Galerkin projection of the momentum equations on the coarse scales, for model reduction. Proven consistency and accuracy properties make this method attractive for tackling transient dynamics problems.

BIOGRAPHICAL SKETCH

Gabriel de Frías hails from Santo Domingo, Dominican Republic, where he attended the Colegio Calasanz for 14 years, until graduating in 2003 as class valedictorian. In 2007 he graduated Summa Cum Laude from the Insituto Tecnológico de Santo Domingo with a bachelor's degree in Civil Engineering. That same year, he obtained the Sloan Fellowship to pursue MS/PhD studies in Structural Engineering at Cornell University. During his graduate studies he interned at Sandia National Laboratories in Albuquerque, New Mexico, as well as serving as a visiting graduate student at Duke University in Durham, North Carolina. Gabriel accepted a position to work for Sandia National Laboratories in Livermore, California upon graduation.

Para mami y papi.

ACKNOWLEDGEMENTS

I want to start by thanking my advisor, Prof. Wilkins Aquino, for his guidance throughout this process. His acumen, patience and dedication were cardinal for my development on a professional and personal level. I would like to thank my committee members, Prof. Anthony Ingraffea, Prof. Timothy Healey and Dr. Martin Heinstein, for providing me with great insight and advice. I am also grateful for my fellow students and research group members, who inspired many productive discussions during my time at Cornell University. This work was made possible thanks to the support of the Computational Solid Mechanics and Structural Dynamics department at Sandia National Laboratories. Dr. Kendall Pierson and the solid mechanics team offered me a breath of technical knowledge and support for which I am forever grateful. Finally I would like to acknowledge the support of the Sloan Foundation, who funded my first three years at Cornell.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The class of problems in which multiple spatial or temporal scales considerably influence the overall behavior of the solution are called "multiscale problems". For them, to efficiently obtain accurate numerical results still presents a major challenge in computational modeling. This is due to the fact that considerable computational costs are associated with representing in detail each of these scales. One could certainly make the case that most problems in science and engineering are multiscale. They have been studied in a wide variety of different fields and applications, like fluid dynamics, bioengineering, failure modeling, weather simulations, population dynamics, fracture mechanics, among many others. With the rapid development of larger and faster computers, the possibility of modeling multiscale problems in detail has become more of a reality.

Over the years, explicit time integration methods for transient dynamics have been widely used. This integration technique is well suited for problems involving large number of degrees of freedom, since it circumvents the need to solve a system of equations at each time step of the analysis. However, an important drawback is that they are conditionally stable, which generally leads to very small "critical" time steps. In this work, we look at ways to reduce the computational time of explicit dynamics simulations by designing and implementing scale decomposition methods that allow us to accurately integrate at time steps larger than the critical one.

The purpose of this work is to study, design and implement novel approaches to address some of the computational challenges of modeling multi-

scale solid mechanics problems with explicit time integration. In structures and solids, is common to have problems where small local features have tremendous impact on the larger scales. Additionally, we are concerned with improving computational performance, while maintaining accuracy. It is important to note that obtaining an explicit description of the behavior of the coarse scales might not be a trivial task. This is where Proper Orthogonal Decomposition (POD) can help. POD is a technique that produces an optimal low dimensional representation of ensembles of high dimensional data. This makes it an attractive technique for describing coarse scale behavior from fine scale computational data.

Two different approaches for addressing multiscale solid mechanics problems are presented in this thesis. Chapter (2) proposes a multiscale mass scaling method that uses POD to eventually obtain larger stable integration time steps, through selective mass scaling, with the purpose of improving performance and maintaining accuracy. Chapter (3) focuses on the application of equation-free/POD techniques, that circumvent some of the drawbacks of standard Galerkin projection of fine scale equations onto the coarse scales. Basic concepts of computational solid mechanics and the finite element method are assumed throughout this work. For the sake of clarity not all of these details will be presented.

CHAPTER 2

MULTISCALE EXPLICIT DYNAMICS METHOD FOR SOLID MECHANICS USING PROPER ORTHOGONAL DECOMPOSITION

Modeling physical systems that span multiple spatial and temporal scales is still one of the main challenges in computational mechanics. Even with the advent of fast supercomputers, it is still computationally demanding to incorporate very small and localized details into a large model. Many times, these local details are of utmost importance to understand the failure mechanism of a much larger structure. In the context of finite elements, the most straightforward approach to incorporate these details, is to perform mesh refinement around the area of interest. Unfortunately, this comes at a great computational cost.

Explicit time integration techniques have been widely used for transient dynamics problems. They are especially attractive for problems involving large number of degrees of freedom and nonlinearities, since a large system of equations does not have to be solved at every time step of the analysis. The main drawback is that these explicit schemes are conditionally stable [9], and therefore, a critical time step is needed to ensure stability. In solid mechanics problems, this critical time step is often very small, and a large number of steps are required to obtain an accurate solution. Furthermore, in finite element simulations, the size of the critical time step is proportional to the size of the smallest element in the mesh. Consequently, using mesh refinement to resolve small geometric details in large models leads to increased computational cost both due to the increased size of the computational mesh, and due to the requirement that a smaller time step be taken. The combination of these two effects can make such models prohibitively expensive to solve, even on massively parallel computers.

Over the years, various techniques [31, 25, 13, 12, 27] have been used to estimate the size of this critical time step. The element-based method [13] has been successfully used, but has shown to provide fairly conservative time step estimates. The Lanczos method [27] provides larger critical time steps, but requires the solution of an eigenvalue problem at multiple stages during the simulation. Some studies [24] have addressed the issue of the added cost of using the Lanczos method and compared it to the cost of the element-based one. Heinstein, Mello and Dohrmann [17] also developed a less conservative, nodal-based, time step estimate that has been heavily used in the SIERRA Mechanics [10] software developed at Sandia National Laboratories.

A wide variety of problems in solid mechanics require high level of spatial mesh refinement (for instance, to capture material or geometric features), but low frequency dynamics dominate the overall response. In this class of problems, time steps much larger than the critical one would be sufficient to accurately integrate the low-frequency dynamics. The need to address these problems led to the development of the multiscale explicit dynamics method proposed herein[39]. The main idea behind the methodology proposed in this thesis, is to decompose the dynamics of the problem into low and high frequency responses captured by coarse and fine spaces, respectively. Then, the concept of mass scaling is applied to the response associated with the high frequencies. The classical concept of mass scaling consists in adjusting the mass of stiff regions in a finite element mesh to allow for a higher critical time step and reduce computational cost [4]. The drawback of this classical approach is that significant errors may arise when the regions where mass scaling is applied have a significant contribution to the overall response. By splitting the response into low and high frequencies, and assuming that the low frequencies dominate

the overall response, it is expected that selectively applying mass scaling only to the high frequency components would yield a more robust method than the classical mass scaling approach.

A very important component of the method presented herein, is the construction of the coarse and fine spaces for the solution approximation. To this end, one approach is to build these spaces geometrically [39]. That is, fine and coarse finite element meshes are used to approximate the high and low frequency responses, respectively. In this approach, the critical time step is governed by the smallest element of the coarse mesh, having the potential for significant savings in computational time, depending on the size of the element in this coarse mesh. However, selecting an adequate coarse mesh *a priori* may be difficult and impractical, especially for problems involving complicated domains. Another, potentially more robust, approach is to build the coarse space using the proper orthogonal decomposition (POD) technique, thus avoiding the use of a coarse mesh.

POD is a technique that has been widely used for model reduction in multiple areas of engineering and physics[1, 26, 5, 6, 33, 29, 14, 20]. The goal of POD is to obtain an optimal low-dimensional representation, or POD modes, of an ensemble of higher dimensional data (experimental or numerical). These modes are directly related to the kinetic energy of the system [5]. Hence the few most energetic modes can be easily determined. Also, POD modes can be obtained from any given ensemble of functions, so this technique makes no assumptions about the linearity of a problem. For these reasons, POD modes are good candidates for building the coarse space in the multiscale explicit dynamics method proposed herein. It is important to point out that POD has been used in a similar

context in the work of Sirisup *et. al.* [37].

The content of this chapter is organized as follows. In Section (2.1) essentials of the mathematical formulation for POD are provided. In Section (2.2), the basic equations for solid mechanics are presented, followed by a detailed description of the proposed multiscale explicit dynamics method using POD. Numerical results are provided in Section (2.3), using two solid mechanics examples, and focusing on showcasing the performance and potential of the method. Finally, Section (2.5) contains a summary of this chapter, and discusses the remaining open issues and future directions of this work.

2.1 Background

In this section, we present the fundamentals of POD. Similar and more detailed presentations on this subject can be found in references [2, 18], but for the sake of completeness the main equations will be provided herein. For all the derivations presented in this work, vectors and tensors will be represented with bold letters and scalars with italic letters.

2.1.1 Proper orthogonal decomposition

We will consider in our derivations the Hilbert space L_2 defined as

$$L_2 = \left\{ \boldsymbol{u} : \int_\Omega |\boldsymbol{u}|^2 d\Omega < \infty,\ \Omega \subset \mathbb{R}^3 \right\}, \tag{2.1}$$

with inner product $(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v}\, d\Omega$, and norm $\|\boldsymbol{u}\| = (\boldsymbol{u}, \boldsymbol{u})^{\frac{1}{2}}$, for all $\boldsymbol{u}, \boldsymbol{v} \in L_2$. If $\boldsymbol{u}$ and $\boldsymbol{v}$ are vectors of length k, then $\boldsymbol{u} \cdot \boldsymbol{v} = \sum_i u_i v_i$.

Given an ensemble of functions $\{\boldsymbol{u}_k\}_{k=1}^n \in L_2$, the main idea behind POD is to find a subspace such that the average distance between this subspace and the ensemble of functions is minimal. Consider the subspace $V^m = \text{span}\{\boldsymbol{\phi}_j\}_{j=1}^m,\ V^m \subset L_2$. POD seeks finite dimensional representations in V^m of the form

$$\hat{\boldsymbol{u}}(\boldsymbol{x}) = \sum_{j=1}^{m} a_j \boldsymbol{\phi}_j(\boldsymbol{x}) \tag{2.2}$$

where $\boldsymbol{\phi}_j(\boldsymbol{x})$ denotes the j^{th} POD mode and a_j is a scalar coefficient. Now, consider a function $\boldsymbol{u}_k \in L_2$ (*i.e.* member of the ensemble). The element $\hat{\boldsymbol{u}}_k^* \in V^m$ that is closest to $\boldsymbol{u}_k$ is defined as

$$\|\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k^*\| \leq \|\boldsymbol{u}_k - \boldsymbol{v}\| \quad \forall \boldsymbol{v} \in V^m, \tag{2.3}$$

Assuming that $\{\boldsymbol{\phi}_j\}$ forms an orthogonal basis, this best approximation is computed as [30]

$$\hat{\boldsymbol{u}}_k^* = \sum_{i=1}^{m} \frac{(\boldsymbol{u}_k, \boldsymbol{\phi}_i)}{\|\boldsymbol{\phi}_i\|^2} \boldsymbol{\phi}_i. \tag{2.4}$$

The goal now is to construct a subspace V^m which is closest, in an average sense, to the ensemble of functions $\{\boldsymbol{u}_k\}_{k=1}^n$. This subspace can be found by solving

$$\underset{V^m \subset L_2}{\text{minimize}} \langle \|\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k^*\|^2 \rangle \text{ s.t. } \|\boldsymbol{\phi}_i\| = 1 \quad (i = 1, ..., m), \tag{2.5}$$

where the averaging operation is defined as

$$\langle \boldsymbol{u}_k \rangle = \frac{1}{m} \sum_{k=1}^{m} \boldsymbol{u}_k. \tag{2.6}$$

It can be shown (see for instance [2, 18]) that the minimization problem described in Eq. (2.5) leads to the following eigenvalue problem

$$\int_\Omega \langle \boldsymbol{u}_k(\boldsymbol{x}) \boldsymbol{u}_k(\boldsymbol{\xi}) \rangle \boldsymbol{\phi}_i(\boldsymbol{x}) d\boldsymbol{x} = \lambda \boldsymbol{\phi}_i(\boldsymbol{\xi}). \tag{2.7}$$

Direct discretization of (2.7) can lead to a very large and computationally expensive eigenvalue problem. The common approach to circumvent this challenge

is to use the method of snapshots [38] in which the eigenvalue problem is transformed into a smaller one given as

$$\frac{1}{m}\sum_{k=1}^{m} A_{jk} d_k^r = \lambda_r d_j^r, \quad A_{jk} = \int_\Omega \boldsymbol{u}_j(\boldsymbol{\xi})\boldsymbol{u}_k(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{2.8}$$

where $[A]_{jk}$ is the correlation matrix whose r^{th} eigenvalue and eigenvector are defined by λ_r and $\boldsymbol{d}^r$, respectively. Once the above eigenvalue problem is solved, the r^{th} proper orthogonal mode is computed as

$$\boldsymbol{\phi}_r(\boldsymbol{x}) = \frac{1}{\lambda_r m}\sum_{k=1}^{m} \boldsymbol{u}_k(\boldsymbol{x}) d_k^r. \tag{2.9}$$

2.1.2 Solid mechanics equations

The initial boundary value problem (IBVP) describing the infinitesimal deformation of a solid body over time is given as

$$\nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b} = \rho \ddot{\boldsymbol{u}} \text{ in } \Omega \tag{2.10a}$$

$$\boldsymbol{\sigma}\boldsymbol{n} = \boldsymbol{\tau} \text{ on } \Gamma_\tau \tag{2.10b}$$

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \text{ on } \Gamma_u \tag{2.10c}$$

$$\dot{\boldsymbol{u}}(\boldsymbol{x}, 0) = \boldsymbol{v}_0(\boldsymbol{x}) \tag{2.10d}$$

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}) \tag{2.10e}$$

where Ω represents the physical domain with boundary Γ, ρ is the density of the material, $\boldsymbol{b}$ is a body force, $\ddot{\boldsymbol{u}}$ is the acceleration, $\boldsymbol{u}$ is the displacement, and $\boldsymbol{\tau}$ represents traction. The part of the boundary where essential conditions are specified is denoted as Γ_u, while the part of the boundary where natural conditions are specified is denoted as Γ_τ. The unit normal vector on the boundary is

denoted as $\boldsymbol{n}$. Furthermore, $\Gamma_u \cap \Gamma_t = \emptyset$ and $\Gamma = \Gamma_u \cup \Gamma_t$. The initial displacement and velocity fields are denoted as $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$, respectively.

It is assumed that the Cauchy stress $\boldsymbol{\sigma}$ is given from a constitutive equation that relates the stress to the strain $\boldsymbol{\varepsilon}$ and its rates as

$$\boldsymbol{\sigma}(\boldsymbol{x}, t) = \boldsymbol{F}\left(\boldsymbol{\varepsilon}(\boldsymbol{x}, t), \frac{\partial \boldsymbol{\varepsilon}(\boldsymbol{x}, t)}{\partial t}, ..., \frac{\partial^n \boldsymbol{\varepsilon}(\boldsymbol{x}, t)}{\partial t^n}\right), \tag{2.11}$$

where the specific form of $\boldsymbol{F}$ is determined by the material model used. The variational form of the problem defined in (2.10) is given as find $\boldsymbol{u} \in \mathcal{U}$ such that

$$\int_\Omega \nabla \boldsymbol{w} : \boldsymbol{\sigma} d\Omega - \int_\Omega \boldsymbol{w} \cdot \rho \boldsymbol{b} d\Omega - \int_{\Gamma_\tau} \boldsymbol{w} \cdot \boldsymbol{\tau} d\Gamma_\tau + \int_\Omega \boldsymbol{w} \cdot \rho \ddot{\boldsymbol{u}} d\Omega = 0 \; \forall \boldsymbol{w} \in \mathcal{W} \tag{2.12}$$

where $\mathcal{U}$ and $\mathcal{W}$ are appropriate function spaces for the trial and test functions, respectively. Using a finite element discretization, the trial and test functions are approximated, respectively, as

$$\boldsymbol{u}_h = \sum_{b=1}^{n} N_b \boldsymbol{u}_b, \quad \text{and} \quad \boldsymbol{w}_h = \sum_{a=1}^{n} N_a \boldsymbol{w}_a \tag{2.13}$$

Substituting these approximations into Eq. (2.12), using the arbitrariness of the test functions, and employing conventional Voigt notation, the weak form is given as

$$\int_\Omega [B]^T \boldsymbol{\sigma} d\Omega - \int_\Omega [N]^T \rho \boldsymbol{b} d\Omega - \int_{\Gamma_\tau} [N]^T \boldsymbol{\tau} d\Gamma_\tau + \int_\Omega \rho [N]^T [N] d\Omega \{\ddot{u}\} = 0, \tag{2.14}$$

where $[N]$ is a matrix representation of the shape functions, $[B]$ contains derivatives of the shape functions, and $\{\ddot{u}\}$ are nodal accelerations.

Defining the internal force vector as

$$\{f_{int}\} = \int_\Omega [B]^T \boldsymbol{\sigma} d\Omega \tag{2.15}$$

the external force vector as

$$\{f_{ext}\} = \int_{\Omega} [N]^T \rho \boldsymbol{b} d\Omega + \int_{\Gamma_\tau} [N]^T \boldsymbol{\tau} d\Gamma_\tau, \tag{2.16}$$

and the mass matrix as

$$[M] = \int_{\Omega} \rho [N]^T [N] d\Omega, \tag{2.17}$$

we obtain a vector form of the semi-discrete system of equations as

$$[M]\{\ddot{u}\} = \{f_{ext}\} - \{f_{int}\}. \tag{2.18}$$

For the sake of compactness in the notation, we will group the external and internal force vectors as

$$\{f\} = \{f_{ext}\} - \{f_{int}\} \tag{2.19}$$

Then, the semi-discrete system can then be expressed as

$$[M]\{\ddot{u}\} = \{f\}, \tag{2.20}$$

which will be the starting point for deriving our proposed multiscale explicit dynamics method.

2.2 Formulation

2.2.1 Multiscale mass scaling method

The multiscale explicit dynamics method presented herein is designed on the assumptions that a clear separation of fine and coarse scales exists, and that the response of the system is dominated by the coarse scales. The main goal of our approach is to integrate the semi-discrete system of equations in (2.20) using

a much larger time step than the stable one required by conventional explicit algorithms, without compromising accuracy. To this end, the main components of the proposed strategy are devising a suitable decomposition of the problem into coarse and fine scales, and constructing an integration scheme that takes advantage of the dominance of the response by the coarse scales.

The initial step is to decompose the discrete accelerations into coarse and fine scale components. Consider the acceleration vector $\{\ddot{u}\}$ to be in a finite dimensional space $U^n = U_f^n \oplus U_c^n$, where U_f^n and U_c^n represent the fine scale space and coarse scale space, respectively. Then, accelerations can be represented as

$$\{\ddot{u}\} = \{\ddot{u}^c\} + \{\ddot{u}^f\},\ \{\ddot{u}^c\} \in U_c^n\ ,\{\ddot{u}^f\} \in U_f^n, \tag{2.21}$$

where the dominant low frequency accelerations are captured in $\{\ddot{u}^c\}$, and the high frequency ones are contained in $\{\ddot{u}^f\}$. Assume that $U_c^n = \text{span}(\{\phi\}_i)_{i=1}^m$. Then, the coarse scale accelerations can be represented as

$$\{\ddot{u}^c\} = [\Phi]\{\ddot{q}\}, \tag{2.22}$$

where $[\Phi]$ represents a matrix whose columns are the vectors $\{\phi\}_i$ and $\{\ddot{q}\}$ is a vector of coefficients whose dimension is that of the coarse space. The number of columns of matrix $[\Phi]$ is determined by the dimension of the coarse space, and the number of rows, typically much larger, is of the same dimension as $\{\ddot{u}\}$. This matrix $[\Phi]$ serves as a prolongation from the coarse space to the fine space.

The next step is to obtain a representation of the fine scale accelerations in terms of the coarse scale accelerations. Using Eq. (2.21), we can express the semi-discrete governing equations as

$$\{f\} = [M]\{\ddot{u}^c\} + [M]\{\ddot{u}^f\}. \tag{2.23}$$

An expression for the coarse accelerations can be obtained using a Galerkin approach. That is, finding the coarse accelerations that satisfy

$$[\Phi]^T(\{f\} - [M]\{\ddot{u}^c\}) = 0. \tag{2.24}$$

Notice that Eq. (2.24) is equivalent to the following orthogonality condition between the coarse and fine scale accelerations.

$$\{\ddot{u}^c\}^T[M]\{\ddot{u}^f\} = 0, \ \forall \ \{\ddot{u}^c\} \in U_c^n, \{\ddot{u}_f^n\} \in U_f^n. \tag{2.25}$$

Defining a matrix $[M_c] = [\Phi]^T[M][\Phi]$ (*i.e.* projection of the mass matrix onto the coarse space), and substituting Eq. (2.22) into Eq. (2.24), we get

$$[\Phi]^T\{f\} - [M_c]\{\ddot{q}\} = \{0\}, \tag{2.26}$$

from which we can obtain the coarse scale acceleration coefficients as

$$\{\ddot{q}\} = [M_c]^{-1}[\Phi]^T\{f\}. \tag{2.27}$$

Then, the coarse scale acceleration vector can be written as

$$\{\ddot{u}^c\} = [\Phi][M_c]^{-1}[\Phi]^T\{f\}. \tag{2.28}$$

From the semi-discrete governing equations and Eq. (2.28), the fine scale component of the inertial force can be computed as

$$\begin{aligned}
[M]\{\ddot{u}^f\} &= \{f\} - [M]\{\ddot{u}^c\} \\
&= \{f\} - [M][\Phi][M_c]^{-1}[\Phi]^T\{f\} \\
&= ([I] - [M][\Phi][M_c]^{-1}[\Phi]^T)\{f\} \\
&= [P]\{f\},
\end{aligned} \tag{2.29}$$

where $[P] = [I] - [M][\Phi][M_c]^{-1}[\Phi]^T$ is an orthogonal matrix projection.

Notice that if $[\Phi]$ is given, the right hand side of Eq. (2.29) can be readily computed within the context of conventional explicit time stepping algorithms. The next step in the formulation is to scale the mass matrix in Eq. (2.29) as to increase the critical time increment. In this work, we will use a central difference algorithm for time integration [4]. To this end, we introduce a modified mass matrix defined as

$$[\widetilde{M}] = [\alpha][M]. \tag{2.30}$$

where $[\alpha]$ is a diagonal scaling matrix whose components are defined as

$$\alpha_i = \begin{cases} \frac{(\Delta t)^2}{4}\left(\frac{\hat{K}_i}{M_i}\right)_{\text{max over i}} & \text{if } \Delta t > \Delta t_c \\ 1 & \text{otherwise.} \end{cases} \tag{2.31}$$

The fine scale accelerations are then computed from Eq. (2.29) as

$$\{\ddot{u}^f\} = [\widetilde{M}]^{-1}[P]\{f\}. \tag{2.32}$$

Substituting Eqs. (2.28) and (2.32) into Eq. (2.21), the total acceleration vector can be obtained as

$$\{\ddot{u}\} = [\widetilde{M}]^{-1}[P]\{f\} + [\Phi][M_c]^{-1}[\Phi]^T\{f\}. \tag{2.33}$$

In Eq. (2.31), Δt is a user-defined time step, $\hat{K}_i$ is an element nodal stiffness, and M_i is the element lumped mass at Node i. The expression for the scaling factor in Eq. (2.31) is based on the nodal-based time step found by Heinstein, Mello and Dohrmann [17], which is less conservative than traditional element-based estimates. Further details of the derivation of this nodal-based time step can be found in Appendix (A). Also, the reader is encouraged to see Reference [17] for additional information on the subject.

Notice that if the user-defined time step Δt is smaller than the stable time step Δt_c, there will be no mass scaling. Indeed, if the latter is the case, Eq. (2.33) is equivalent to the original semi-discrete system (2.20). Otherwise, the mass associated with the fine scale accelerations is scaled as to render the integration scheme stable with respect to the given time step. This is one of the key differences between our approach and model reduction approaches as the one used in [26]. That is, by the virtue of the decomposition into fine and coarse spaces, the integrated displacement field always has contributions from both the fine and coarse scales. Furthermore, by selecting $\Delta t \leq \Delta t_c$, we can always recover the original accuracy and stability of the central difference algorithm, regardless of the coarse space approximation properties.

Lastly, we would like to point out that the user defined time step cannot be arbitrarily large as the time integration algorithm is still conditionally stable. For instance, large mass scaling of the high frequency accelerations may lead to a critical time step controlled by the largest frequency of the coarse scale accelerations. A precise definition of a new stable time step in the context of the proposed multi scale formulation is outside the scope of the current work, but will be pursued in a future investigation.

2.2.2 Computing the POD modes and construction of the coarse space

In this section, we present a method for efficiently constructing the coarse space matrix $[\Phi]$. In previous work [39], this matrix was built geometrically using coarse finite element meshes. The advantage of the latter approach is that a new,

larger stable time step for the simulation can be easily obtained by computing the stability limit of the coarse grid. However, selecting an adequate coarse grid (*a priori*) in terms of accuracy and speed can be very challenging in realistic simulations. For instance, coarse grids could be inadequate to capture localized features of a solution.

We propose using POD to construct the coarse space in order to circumvent some of the drawbacks associated with geometric approaches. POD modes are excellent candidates as a basis for the coarse space because of their optimal representation properties. Moreover, the most dominant POD modes are easily determined by sorting the eigenvalues of the covariance matrix. The latter fact provides sound guidance for the selection of an adequate coarse space. It is important to point out that POD modes have been used for similar multiscale decompositions in the work of Sirisup et al. [37]. These authors used POD modes within an equation-free framework to obtain fast and accurate solutions to multiscale incompressible flow problems.

In the spirit of the discrete formulation shown in Section (2.2.1), all functions in the eigenvalue problem shown in Eq. (2.8) are discretized using finite elements. Hence, POD modes are interpolated as

$$\boldsymbol{\phi}_r^h(\boldsymbol{x}) = \sum_j N_j(\boldsymbol{x})\phi_{rj} \tag{2.34}$$

where ϕ_{rj} represents the value of mode r at Node j. The nodal modes $\{\phi\}_r$ are then used as a basis for the coarse space. One attractive advantage of POD modes is that in the case of a constant density, the projected mass matrix is simply obtained as

$$[M_c] = [\Phi]^T[M][\Phi] = \rho[I]. \tag{2.35}$$

The above relationship arises from the orthonormality of the POD modes. That

is,

$$\{\phi\}_i^T \int_\Omega [N]^T [N] d\Omega \{\phi\}_j = \delta_{ij}$$

where δ_{ij} is the Kronecker's delta.

In this work, the ensemble of functions needed to compute the POD modes were taken as the displacements obtained at early time steps of the full scale simulation. That is, displacement fields obtained using a central difference algorithm with time steps below the critical one and sampled at prescribed times. Ideally, the simulation time used for building the POD modes would be much shorter than the total time required for the given problem. After the desired snapshots have been obtained, the POD modes are computed, and the multiscale approach proposed herein is carried out. Figure (2.1) shows a sketch of this process. The gray area in the figure represents the part where the full scale simulation is performed at or below the critical time step, Δt_c, up to a time t_{fs}. The data ensemble is constructed by sampling displacements obtained within the time interval $(0, t_{fs})$. The POD modes are then constructed using this data ensemble, and time integration is carried out at a time step $\Delta t_{pod} > \Delta t_c$ using from the proposed multiscale approach.

2.2.3 Remarks on consistency and stability

Notice that because our proposed method is based on the central difference algorithm and the multiscale decomposition is used just as a mean to scale the mass, the proposed strategy inherits the properties of central difference algorithms. That is, it is consistent and (conditionally) stable, hence convergent. Furthermore, the rate of convergence (asymptotically) would be that of the cen-

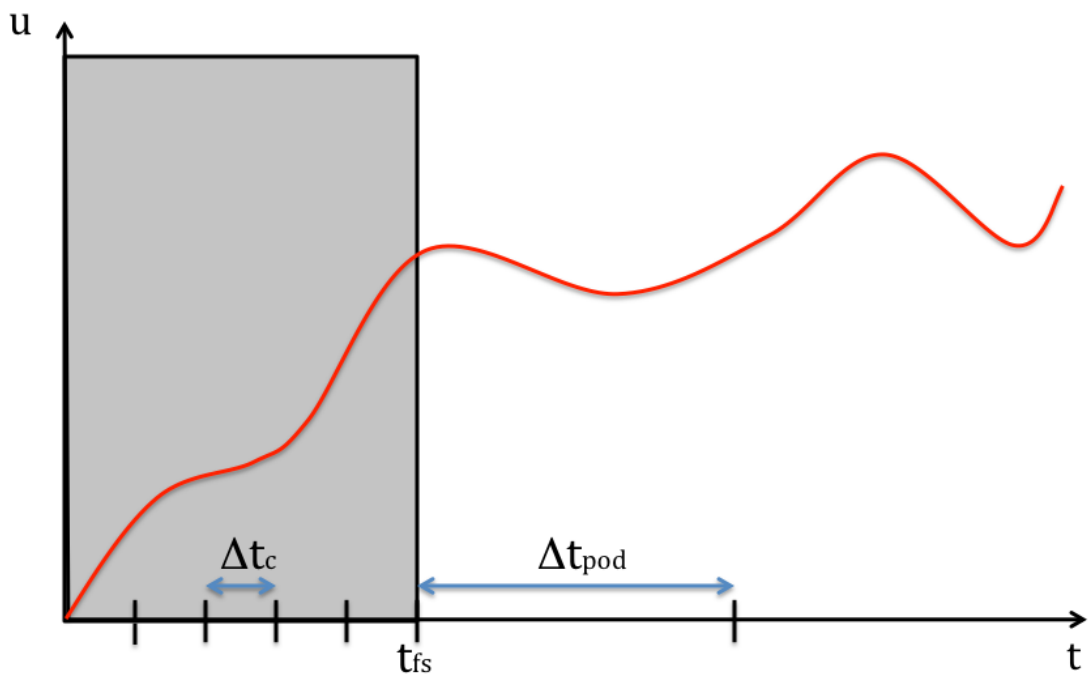


Figure 2.1: Simple sketch of the proposed multiscale explicit dynamics method using POD

tral difference algorithm. Notice that for time steps smaller than the critical one, there is no mass scaling and the proposed approach reduces to the conventional central difference algorithm (see Eqs. (2.30)-(2.33)).

2.2.4 Algorithm flow

The proposed multiscale explicit dynamics method using POD has been implemented within the context of an explicit central difference time integrator. Further details on this explicit time integration scheme can be found in Reference [4]. The main steps involved in the proposed algorithm are outlined next.

1. For $0 \leq t \leq t_{fs}$, perform explicit central difference time integration at or

below a critical element-based time step Δt_c, and gather n snapshots of the nodal displacements $\{u(t)\}$ at times $\{\frac{t_{fs}}{n}, \frac{2t_{fs}}{n}, ..., t_{fs}\}$.

2. Compute POD modes using the data ensemble and Eq. (2.9). Then, build the coarse space matrix $[\Phi]$.

3. Set new time step $\Delta t_{pod} \geq \Delta t_c$.

4. While $t \leq t_f$

   (a) Compute the scaling matrix $[\alpha]$ from Equation (2.31), and compute the scaled mass $[\widetilde{M}]$ using Eq. (2.30).

   (b) Compute the coarse nodal accelerations $\{\ddot{u}^c\}$ from Eq. (2.28).

   (c) Compute the fine scale accelerations $\{\ddot{u}^f\}$ from Eq. (2.32)

   (d) Compute total accelerations $\{\ddot{u}\}$ using Eq. (2.21)

   (e) Update nodal velocities $\{\dot{u}\}$.

   (f) Update nodal displacements $\{u\}$.

   (g) Update nodal forces $\{f\}$.

2.3 Numerical results

It this section, we present several numerical examples that illustrate the performance of the proposed multiscale explicit dynamics method using POD. The objective of the examples is to show how our selective mass scaling leads to improved computational speeds, while maintaining accuracy. These examples are all run twice: once with standard central difference explicit time integration, and once with the multiscale approach proposed here. To this end, we use a

performance improvement factor (PIF) defined as

$$\text{PIF} = \frac{T_{cd}}{T_{ms}}, \tag{2.36}$$

where T_{cd} is the total CPU time of an explicit dynamics simulation that uses an element-based critical time step, and T_{ms} is the total CPU time corresponding to a simulation that uses of our proposed approach.

We also quantified the error between the solution obtained through a pure central difference algorithm using a critical element-based time step, and that obtained with our proposed approach. For this, we use an error defined as

$$e_r = \sqrt{\frac{\sum_{i=1}^{n_t} \|\{u_{cd}\}_i - \{u_{ms}\}_i)\|_{\ell_2}^2}{\sum_{i=1}^{n_t} \|\{u_{cd}\}_i\|_{\ell_2}^2}}, \tag{2.37}$$

where $\{u_{cd}\}_i$ corresponds to the displacement at time t_i obtained using a conventional central difference algorithm, and $\{u_{ms}\}_i$ represents the displacement solution obtained using the proposed multiscale method. The operation $\|.\|_{\ell_2}$ represents the Euclidian norm of a vector. Recall that before time t_{fs} our proposed approach uses a full-scale central difference algorithm to compute the snapshots for building the POD modes. For this reason, the displacements used to calculate the error in Eq. (2.37) were taken after the POD modes were computed, and the proposed algorithm started. That is, for times $t > t_{fs}$. It is important to point out that all the time steps after t_{fs} up to the final time were used for the calculation of this error.

The results used for quantifying the performance of our approach, as shown in all tables of this chapter, were obtained using the 5 most dominant POD modes. These 5 modes accounted for more than 99% of the total sum of all eigenvalues. That is, we used the eigenvalue spectrum computed from Eq. (2.8)

to compute the quantity $k(m)$ defined as

$$k(m) = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{k=1}^{p} \lambda_k} \times 100\%, \tag{2.38}$$

where λ_i is the eigenvalue associated to the i^{th} POD mode, m is the number of POD modes used in the multiscale problem, and p represents the total number of POD modes that were obtained from Eq. (2.8).

2.3.1 Example 1: Linear elastic cylinder with legs

The domain for this example problem consists of a hollow thin steel cylinder with legs. The material in the entire cylinder is assumed to be linearly elastic. Fixed displacement boundary conditions were specified on the bottom surfaces of the legs. A transient pressure load was applied on a small square area of the side of the cylinder as shown in Figure (2.2). The load amplitude was increased linearly from 0 to 10^5 psi for a total simulation time of 5×10^{-4} s. The density and Young's modulus of the material were taken as 7.32×10^{-4} lbf $\cdot$ sec^2/in^4 and 30×10^6 psi, respectively. The mesh used for this problem consisted of 43,308 8-node hexahedral elements, and the corresponding element-based critical time step was 1.23×10^{-8} s.

As mentioned in the previous section, the user can choose when to compute the POD modes and start the proposed multiscale algorithm. Two main scenarios were used in this example problem. In the first scenario, we gathered 10 snapshots using a conventional explicit central difference algorithm and computed the POD modes 5×10^{-5} s into the simulation (10% of the total time). In the

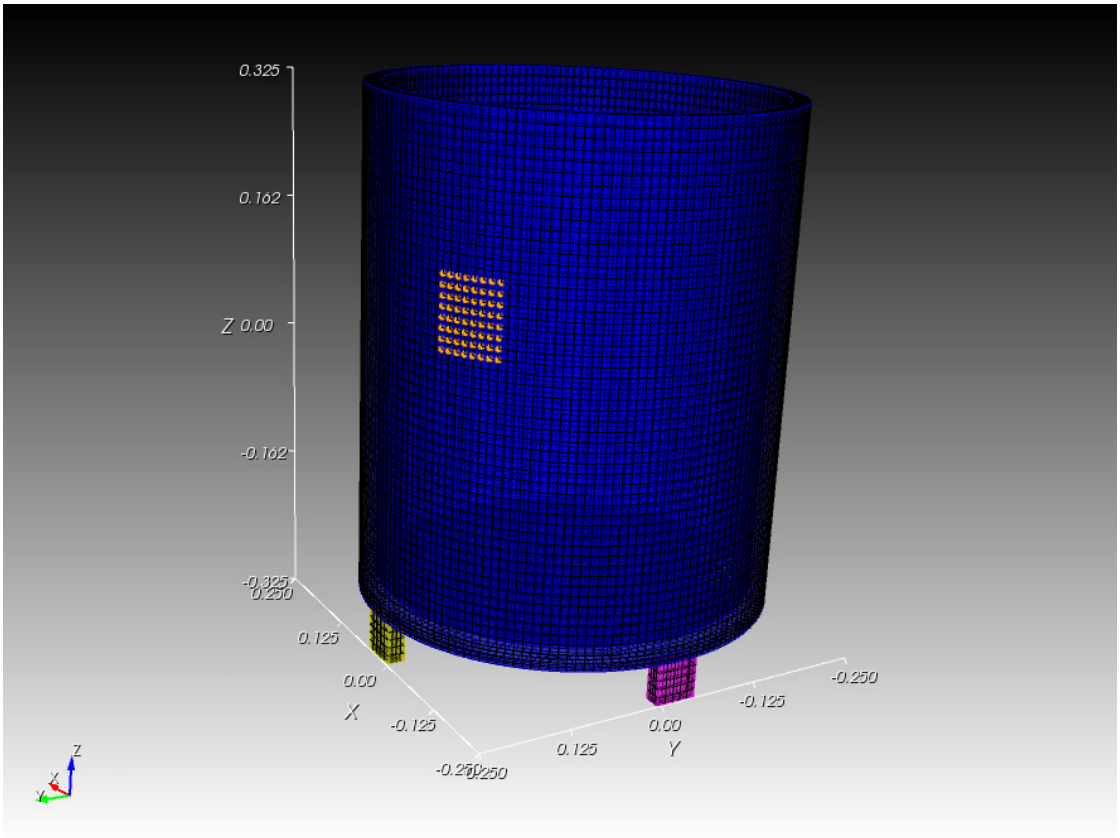
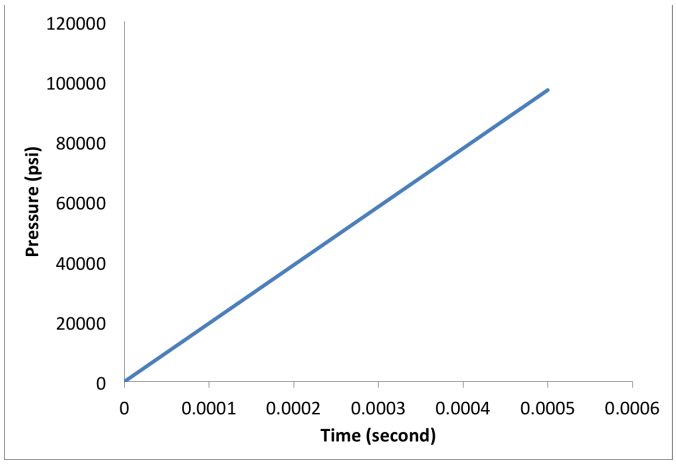


Figure 2.2: Mesh of hollow linear elastic cylinder with legs

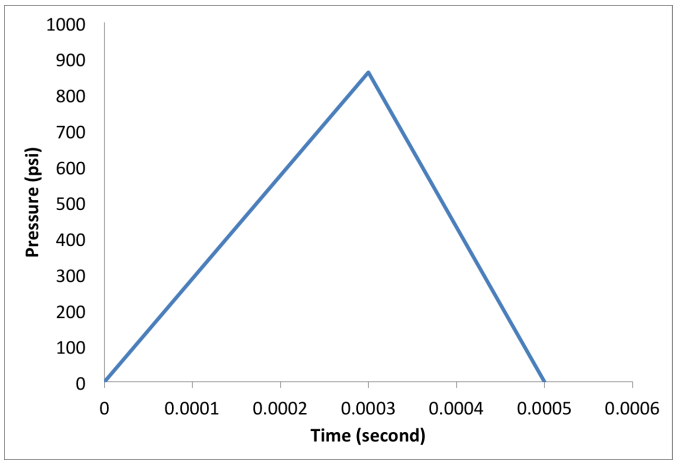
second case, we used 20 snapshots to compute the POD modes at 1×10^{-4} s into the simulation time (*i.e.* 20% of the total time). The rest of the simulation used a coarser, user-defined time step Δt_{pod} as seen in Figure (2.1). For both scenarios, snapshots were taken every 5×10^{-6} s, and displacement were computed using 1, 2, 3, 4, and 5 of the most dominant POD modes to build the coarse matrix $[\Phi]$. These cases were then compared to results obtained from a central difference explicit dynamics algorithm using the critical element-based time step Δt_c. It is important to point out that the main reasoning behind using 20% of the simulation, as well as 10%, is to show that indeed an increase in accuracy can be achieved, by using additional information to build the POD modes.

Figure (2.4) shows contour plots of the displacement magnitude at the end of the simulation obtained using the proposed multiscale approach and a conventional central difference algorithm. We used 20 snapshots and the 5 most dominant POD modes in the results shown for the multiscale method. Furthermore, we used a time step $\Delta t_{pod} = 3.69 \times 10^{-8}$ s after the POD modes were computed, which is equivalent to three times the element-based time step. It can be observed from Figure 2.4, that the multiscale approach produced a final displacement field very close to that obtained using a conventional explicit dynamics algorithm.

We studied the improvement in computational cost and loss of accuracy for the proposed method as compared to a conventional central difference algorithm. Table (2.1(a)) shows the performance improvement factors and errors in the solution for POD modes computed at 20% of the simulation time and using time steps Δt_{pod} of 3, 10 and 20 times larger than the critical element-based one. It can be observed that that we were able to use time steps as large as 20 times the size of the element-based critical time step with just a 2.69% relative error. The corresponding performance improvement factor defined in Eq. (2.36) was 3.56. That is, the total simulation time for the proposed multiscale approach was less than one third that of a conventional central difference algorithm. Table (2.2(a)) shows that when the POD modes were computed using just 10% of the simulation time, a larger performance improvement was achieved, as expected. However, this improved performance came at the expense of a higher relative error, also as expected. The increase in relative error was likely due to the fact that at 10% of the simulation time less information about the eventual dominant dynamics of the system was available in the POD modes as compared to the case when 20% of the simulation time was used. The trade-off between compu-



(a)

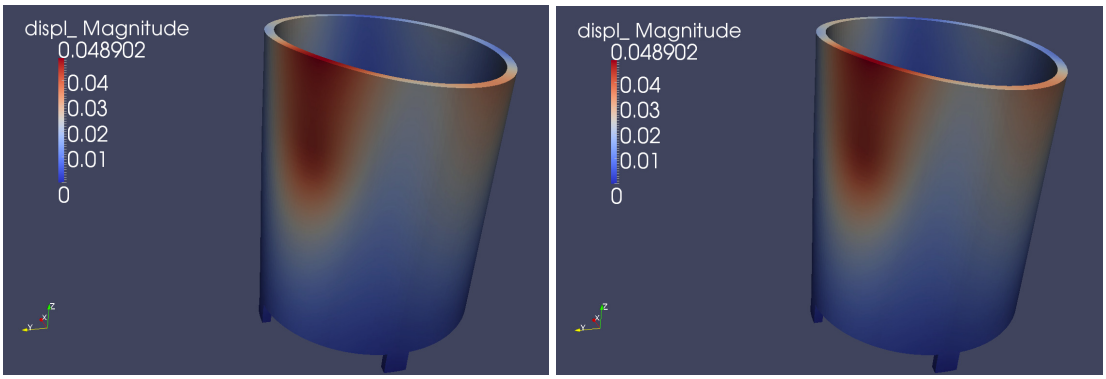


(b)

Figure 2.3: Transient pressure loading: a) used in Example 1; b) used in Example 2.

tational cost and error can be observed, for instance, in Table (2.2(a)) where at 20 times the critical element-based time step the improvement factor was 5.34 and the relative error was 3.41%. In addition, Tables (2.1(a)) and (2.2(a)) also show that relative errors in stress were very similar to those found in displacements.

It is important to point out that only a small percentage of the time, from the initial fine scale computation, was spent computing the POD modes. When using 20% of the total time, the initial central difference calculation took 503.7 s,



(a) (b)

Figure 2.4: Displacement magnitude for Example 1 at a time of 5×10^{-4} s: a) using conventional central difference algorithm; b) using multiscale approach with 20 snapshots and 5 POD modes.

from which only 22.81 s (4.50% of the time) accounted for computing the POD modes. In the case where 10% of the total time was used, the initial central difference calculation took 233.42 s, from which only 6.496 s (2.79% of the time) were spent computing POD modes.

We also investigated the performance of our proposed method when the simulation time used to collect snapshots and build the POD modes is excluded. That is, looking at the simulation period for $t_{fs} < t \leq t_f$. We can see in Tables (2.1(b)) and (2.2(b)) that the improvement factor was significantly higher than when the total simulation time is used, as expected. For example, Table (2.1(b)) shows that for 20 times the critical element-based time step the performance improvement factor was 11.38. Similarly, Table (2.2(b)) shows an improvement factor of about 10 for a time step 20 times larger than the critical one. It is important to realize that performance is improved by a factor of 10, even though the time step was increased by a factor of 20, because of the extra matrix and vector operations involved in computing the coarse and fine scale accelerations.

Table 2.1: Example 1. POD modes computed at 20% of the simulation time

(a) Performance improvements for total simulation time

Method	Time step (s)	CPU time (s)	PIF	Displacement relative error %	Von Mises stress relative error %
Explicit dynamics	Δt_c	2396.29	-	-	
Multiscale explicit	$3(\Delta t_c)$	1300.60	1.84	1.06%	1.14%
dynamics using	$10(\Delta t_c)$	776.90	3.08	2.00%	1.84%
5 POD modes	$20(\Delta t_c)$	673.02	3.56	2.69%	2.95%

(b) Performance improvements after computation of POD modes

Method	Time step (s)	CPU time (s)	Performance improvement factor
Explicit dynamics	Δt_c	1889.26	-
Multiscale explicit	$3(\Delta t_c)$	793.57	2.38
dynamics using	$10(\Delta t_c)$	269.87	7.00
5 POD modes	$20(\Delta t_c)$	165.99	11.38

Figures(2.5) and (2.6) show the effect of the number of POD modes on the accuracy of the computed solution. Furthermore, these figures show how the mass-scaled solution converges to that of the conventional central difference algorithm as the time step is decreased. To this end, the L_2 error defined in Eq. (2.37) was plotted against a time step ratio defined as $T_c = \frac{\Delta t_{pod}}{\Delta t_c}$. Notice that $T_c \leq 1$ means that no mass scaling was performed and the solution corresponded to that of a conventional central difference algorithm (*i.e.* $e_r = 0$). The calculations were performed using 1, 2, 3, and 5 POD modes. As expected, Figures (2.5) and (2.6) show that as the time step ratio decreased, the error de-

Table 2.2: Example 1. POD modes computed at 10% of the simulation time

(a) Performance improvements for total simulation time

Method	Time step (s)	CPU time (s)	PIF	Displacement relative error %	Von Mises Stress relative error %
Explicit dynamics	Δt_c	2396.29	-	-	
Multiscale explicit	$3(\Delta t_c)$	1176.92	2.04	1.68%	1.58%
dynamics using	$10(\Delta t_c)$	582.33	4.11	2.61%	2.24%
5 POD modes	$20(\Delta t_c)$	448.76	5.34	3.41%	3.14%

(b) Performance improvements after computation of POD modes

Method	Time step (s)	CPU time (s)	Performance improvement factor
Explicit dynamics	Δt_c	2162.86	-
Multiscale explicit	$3(\Delta t_c)$	943.50	2.29
dynamics using	$10(\Delta t_c)$	348.91	6.20
5 POD modes	$20(\Delta t_c)$	215.34	10.04

creased towards zero regardless of the number of modes used. For the case when the POD modes were computed at 10% of the simulation time (Figure (2.5)), it can be noticed that the the error decreased as more POD modes were used and the trend held regardless of the time step ratio. Moreover, is is apparent from the figure that there was not much accuracy gained by using more than three POD modes in this example. The latter behavior can be attributed to the fact that the eigenvalues of the first three modes heavily dominated the eigenvalue spectrum. More specifically, we obtained $k(3) > 99.99\%$ from Eq. (2.38). Similar trends are observed for the case when the POD modes were computed

using 20% of the simulation time. Also, in this case it can be observed that three POD modes were adequate for constructing the coarse space and there was no significant advantage in using more modes. Again, the reason for this behavior being the dominance of the eigenvalues corresponding to these modes. It is interesting to observe (See Tables (2.2(a)) and (2.1(a))) that the two cases studied herein produced similar accuracy. Hence for this example, the best performance, in practice, would be obtained by using only 10% of the simulation time to generate the POD modes.

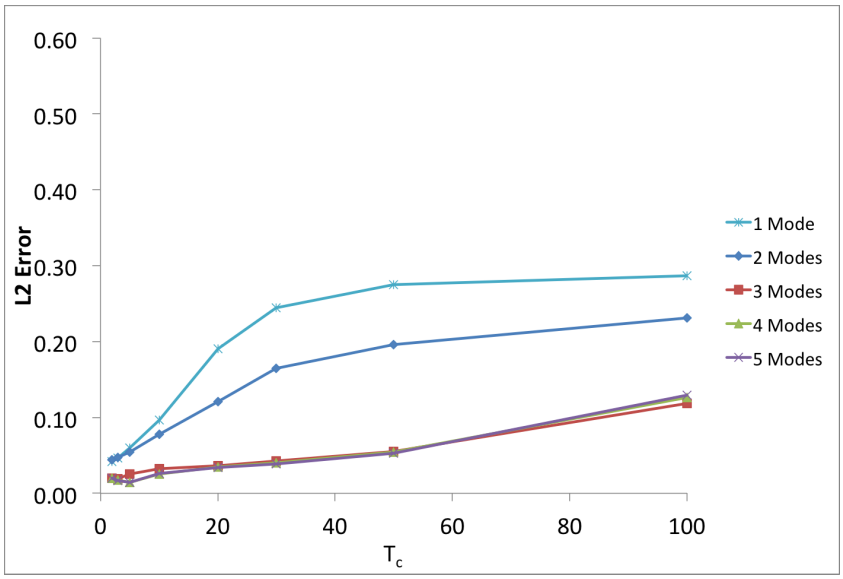


Figure 2.5: Relative error vs. time. POD modes computed at 10% of the run for Example 1.

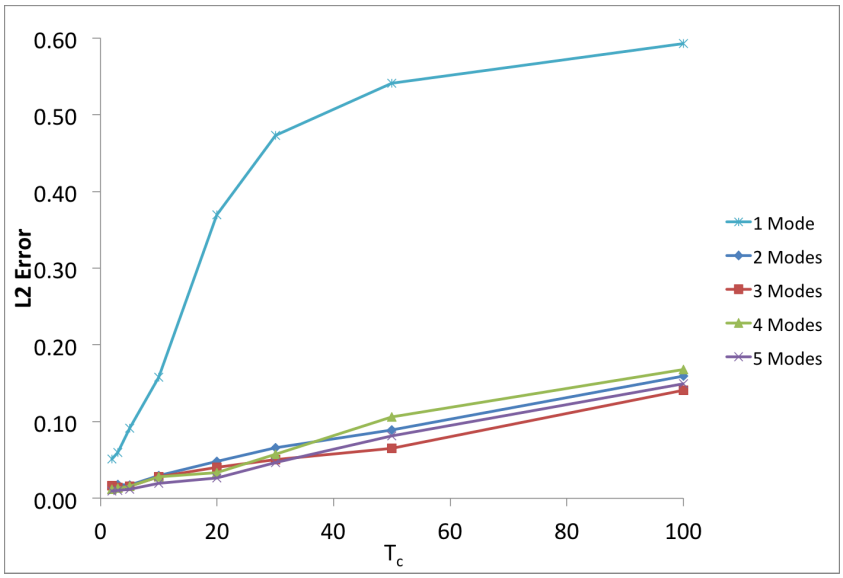


Figure 2.6: Relative error vs. time. POD modes computed at 20% of the run for Example 1.

2.3.2 Example 2: Indentation on hollow cylinder with elastic-plastic material model

The purpose of this example is to assess the performance of the proposed method in nonlinear problems and in the presence of localized deformations. The domain for this problem consisted of a cylinder made of an elastic-plastic material and subjected to a localized force. The bottom of the cylinder had a prescribed fixed displacement boundary condition. As shown in Figure (2.3), the pressure load used in this example increased linearly from 0 to 860 psi in 3×10^{-4} s, and then decreased linearly to 0 until the end of the simulation time (*i.e.* 5×10^{-4} s). The load was applied in the x direction to the region with the highlighted nodes in Figure (2.7). The yield stress for this material was taken as $10,000$ psi, ensuring yielding of the material under the current loading conditions. The density and Young's modulus of the material were set at

2.58×10^{-4} lbf $\cdot$ sec^2/in^4 and 1×10^7 psi, respectively. The mesh used for this problem consisted of 44,640 8-node hexahedral elements and the stable element-based time step was computed to be 1.17×10^{-8} s.

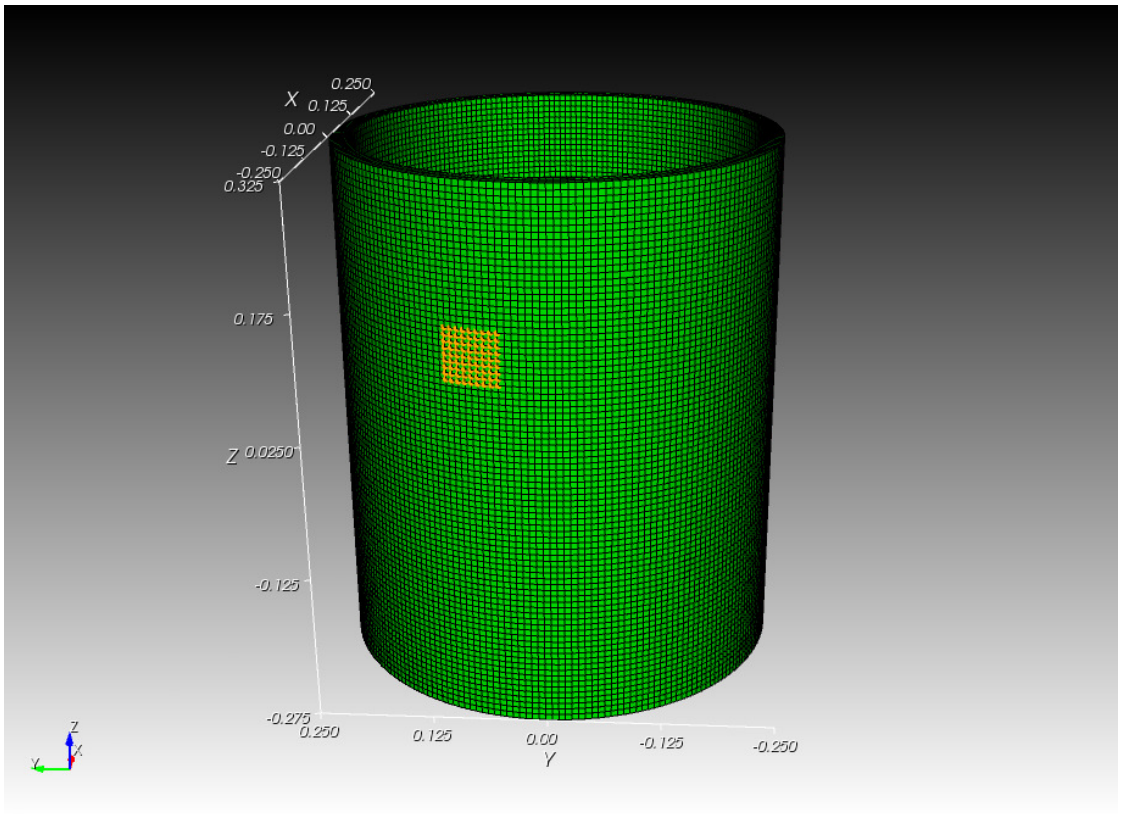


Figure 2.7: Mesh of elastic-plastic aluminum hollow cylinder

As in the previous example, two main scenarios were used in this example problem. For these, snapshots were taken every 5×10^{-6} s, and results were obtained using 1, 2, 3, 4, and 5 of the most dominant POD modes to build the coarse matrix [Φ]. In the first scenario, we gathered 10 snapshots of the solution in the a time span of $5{\times}10^{-5}$ s (10% of the total time), while in the second scenario we computed POD modes using 20 snapshots in a time span of 1×10^4 s (20% of the total time). The rest of the simulation was carried out at a time step Δt_{pod}, which was always larger than the critical element-based time step, Δt_c. The

improvement factors and relative errors obtained obtained for this problem are reported in Tables (2.3) and (2.4). These values were obtained for Δt_{pod} equal to 2, 3, and 5 times larger than Δt_c. Also for this problem, the results of these scenarios were compared to results from a explicit dynamics analysis that used an element-based stable time step.

Table 2.3: Example 2. POD modes computed at 20% of the simulation time

(a) Performance improvements for total simulation time

Method	Time step (s)	CPU time (s)	PIF	Displacement relative error %	Von Mises Stress relative error %
Explicit dynamics	Δt_c	3014.24	-	-	
Multiscale explicit	$2(\Delta t_c)$	2058.03	1.46	7.31%	9.16%
dynamics using	$3(\Delta t_c)$	1641.76	1.84	9.96%	21.90%
5 POD modes	$5(\Delta t_c)$	1206.22	2.50	10.70%	33.56%

(b) Performance improvements after computation of POD modes

Method	Time step (sec)	CPU time (sec)	Performance improvement factor
Explicit dynamics	Δt_c	2421.68	-
Multiscale explicit	$2(\Delta t_c)$	1465.47	1.65
dynamics using	$3(\Delta t_c)$	1049.20	2.31
5 POD modes	$5(\Delta t_c)$	613.67	3.95

Figure (2.8) shows contour plots of the displacement magnitude obtained using a conventional central difference algorithm and with the proposed multiscale approach. For the latter, we used 20 snapshots of the solution and the

Table 2.4: Example 2. POD modes computed at 10% of the simulation time

(a) Performance improvements for total simulation time

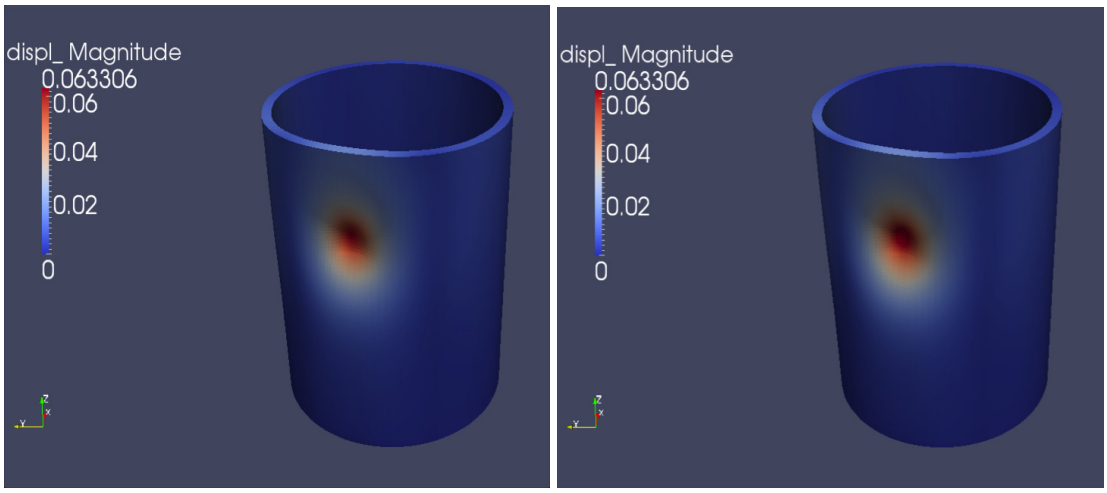
Method	Time step (s)	CPU time (s)	PIF	Displacement relative error %	Von Mises Stress relative error %
Explicit dynamics	Δt_c	3014.24	-	-	
Multiscale explicit	$2(\Delta t_c)$	2135.01	1.41	8.09%	9.46%
dynamics using	$3(\Delta t_c)$	1587.55	1.90	11.14%	23.62%
5 POD modes	$5(\Delta t_c)$	1068.09	2.82	13.57%	35.49%

(b) Performance improvements after computation of POD modes

Method	Time step (sec)	CPU time (sec)	Performance improvement factor
Explicit dynamics	Δt_c	2694.52	-
Multiscale explicit	$2(\Delta t_c)$	1815.29	1.48
dynamics using	$3(\Delta t_c)$	1267.83	2.13
5 POD modes	$5(\Delta t_c)$	748.37	3.60

5 most dominant POD modes. The element-based stable time step was used in the central difference scheme, while for the proposed method we used a $\Delta t_{pod} = 2.34 \times 10^{-8}$, which is twice as large as the element-based stable time step. From Figure (2.8), we can observe that the results obtained at the final time of the simulation using both methods were very similar.

Tables (2.3) and (2.4) show CPU times and performance improvement factors along with the relative errors associated with each simulation. As observed



(a) (b)

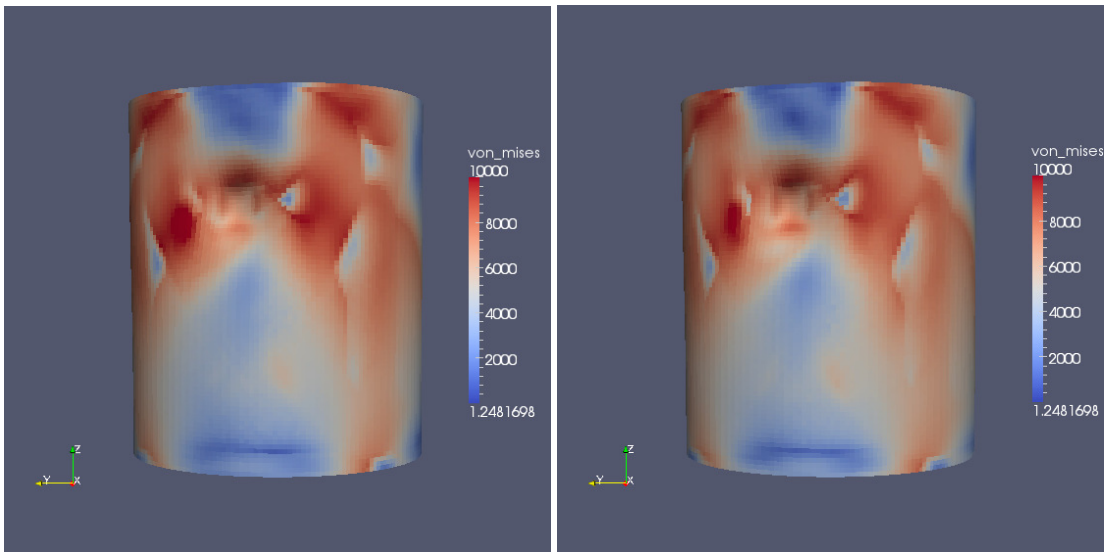
Figure 2.8: Results for example 2 at termination time of 5×10^{-4} s: a) using conventional central difference algorithm; b) using multiscale approach with 20 snapshots and 5 POD modes.

from these tables, the performance improvement and relative errors follow similar trends as those shown in Example 1. That is, the performance improved when fewer snapshots were used to construct the coarse space, but the error increased. The tables also show that the performance improvement factor ranged between 1.4 and 3, which represent significant computational savings in all cases. Despite the fact that the computed errors in stresses (in the ℓ_2 norms used) were noticeably higher than the errors in displacements, we noticed that the spatial distribution of the stress field appeared reasonable throughout all time steps. Contour plots of the Von Mises stress are shown in Figure (2.9) for both the conventional central difference algorithm and the proposed method. In this case, we used only 10% of the simulation time to build the POD modes, and 5 POD modes were used to build the coarse space. A time step twice as large as the element-based one was used. Von Mises stresses are presented at times 2.75×10^{-4} s, and 5×10^{-4}. Figure (2.9) shows that similar Von Mises stress dis-

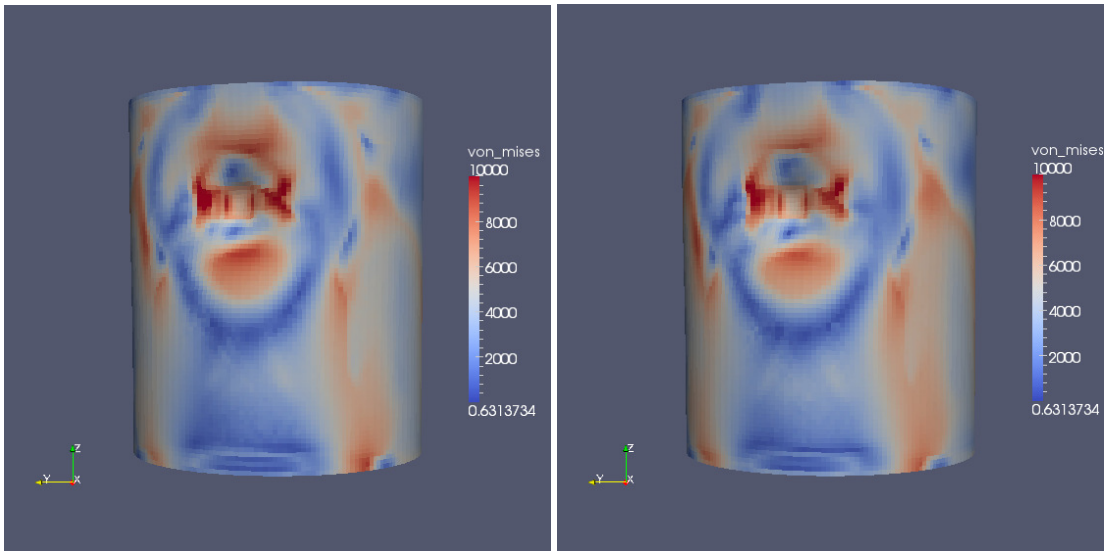
tributions were obtained at these two time frames when using a conventional central difference algorithm and our multiscale approach. We confirmed that this pattern was consistent throughout all the time steps in this simulation.

As in example 1, only a small percentage of the time, from the initial fine scale computation, was spent computing the POD modes. When using 20% of the total time, the initial central difference calculation took 592.56 s, from which 27.535 s (4.65% of the time) accounted for computing the POD modes. In the case where 10% of the total time was used, the initial central difference calculation took 319.72 s, from which only 6.804 s (2.13% of the time) were spent computing POD modes.

Tables (2.3) and (2.4) also show that the relative errors were larger in this example problem than those obtained in the linear elastic case studied in Example 1. This can be attributed to different factors. For instance, the integration of the rate equations in plasticity adds a potential source of error not present in the linear elastic case. Furthermore, It is also likely that more snapshots taken over longer time spans are needed to build an effective coarse space with POD modes. The latter observation is in agreement with the behavior of the error shown in Figures (2.10) and (2.11), where it can be observed that the POD expansion is not as effective in approximating the solution as in the linear elastic case. Furthermore, the need for taking snapshots in a time span larger than 10% of the simulation is reinforced by the fact that there was a marked decrease in the error when 20% of the simulation time was used to sample the snapshots, and 4 or 5 modes were used to build the coarse space. The latter source of error could be, however, easily remedied by using more snapshots over a longer time span. As in Example 1, Figures (2.10) and (2.11) show, in general, that as the



(a) (b)



(c) (d)

Figure 2.9: Von Mises stress results for example 2 using: a) conventional central difference algorithm at 2.75×10^{-4} s; b) multiscale approach using 10 snapshots and 5 POD modes at 2.75×10^{-4} s; c) conventional central difference algorithm at 5×10^{-4} s; d) multiscale approach using 10 snapshots and 5 POD modes at 5×10^{-4} s.

number of POD modes increased, the relative error in the solution decreased. Also as observed in Example 1, there is no significant gain in accuracy beyond

a POD expansion with 4 or 5 modes when 20% of the simulation was used to obtain the modes.

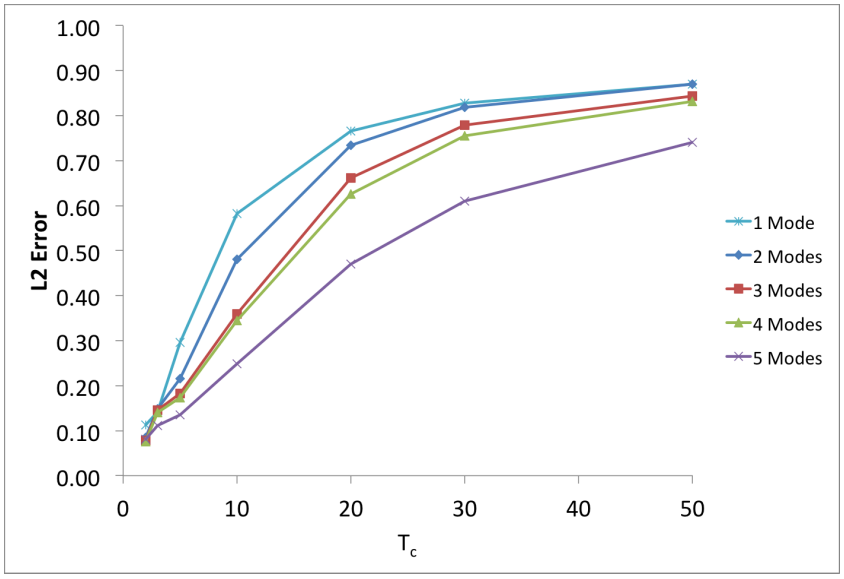


Figure 2.10: Relative error vs. time. POD modes computed at 10% of the run for Example 2.

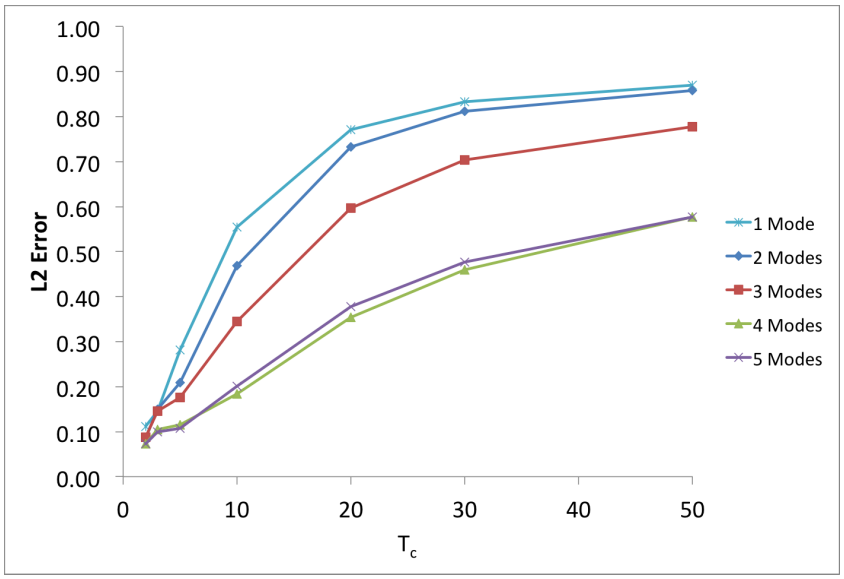


Figure 2.11: Relative error vs. time. POD modes computed at 20% of the run for Example 2.

We have observed that the errors for this nonlinear problem are larger than the ones obtained for the linear elastic case in Example 1. One way to address this is issue can be recomputing the POD modes at multiple steps throughout the simulation. This would ensure that additional information about the dynamics of the system is included in the new POD modes. For this example, after the initial 10% of the modes were used, additional time steps were collected, and modes were rebuilt at every 5×10^{-5} s. In total, POD modes were computed 9 times throughout the simulation. Table V compares the performance improvement factors and relative errors of simulations when POD modes are and are not recomputed. As expected, errors noticeably decreased, but at the cost of some extra computing time. For example, using 3 times the critical time step brought down performance improvement from 1.90 to 1.59, but decreased error from 11.14% to 6.52%. These results confirm that accuracy can be improved by recomputing the POD modes, but the user will have to be mindful that these additional operations will also affect performance.

Table 2.5: Multiscale approach using 5 POD modes and 10% of the simulation time

	Performance Improvement Factor		Displacement relative error %	
Time step Δt_c	POD modes not recomputed	POD modes recomputed	POD modes not recomputed	POD modes recomputed
$2(\Delta t_c)$	1.41	1.12	8.09	5.67
$3(\Delta t_c)$	1.90	1.59	11.14	6.52
$5(\Delta t_c)$	2.82	2.22	13.57	8.23

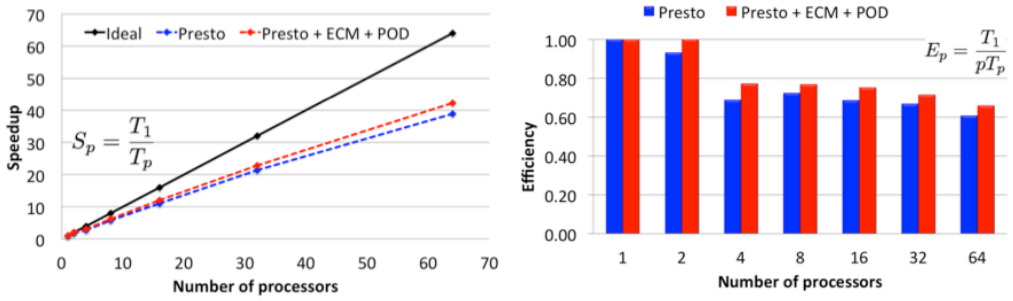
2.4 Implementation notes

The implementation of this algorithm was done in the SIERRA Mechanics code, developed at Sandia National Laboratories. The code consists of a multi-physics suite of analysis modules that address Sandia's scientific applications, such as, structural dynamics, solid mechanics, thermal and fluid analysis, among others. This multiscale mass scaling algorithm was implemented in the SIERRA/SM (formerly Presto)[40] module, which focuses on analyzing 3D structures and solids with both implicit and explicit transient dynamics codes.

For this work, a main Proper Orthogonal Decomposition class was created. All of the functionality needed to obtain data and compute POD modes was implemented, such as collection of snapshots, assembly of correlation matrix, calls to eigenvalue solvers, computation of POD modes, orthogonality checks, snapshot approximation, among others. Additionally, this POD implementation was coupled with the existing explicit dynamics module, in order to implement the necessary multiscale mass scaling method proposed in this chapter. We also gave the user full control of the main variables directly from the input file, such as number of snapshots to be taken, amount of energy the POD modes should account for, time step for POD mode computation, size of coarse time step, frequency of POD mode refresh, etc.

Additionally, SIERRA is designed to be massively parallel, so we extended the implementations of this method to allow parallel computing capabilities. With these additions, our approach is expected to scale similarly to the SIERRA/SM (Presto) module. In order to test this we ran several problems with different number of processors, and computed speedup and efficiency. Speedup

is the ratio between the CPU time for a serial run, and CPU time of the parallel run. Efficiency is a performance metric obtained by the ratio of speedup to number of processors, and is used for describing how well the processors are being used. Figures (2.13) and (2.13) show speedup and efficiency plots for the problem described in Section (2.3.1) with meshes of 346,464 and 2,771,712 8-node hexahedral elements, respectively. Here, the blue lines represent a simulation that used only the explicit dynamics algorithm of Sierra/SM (Presto), and the red lines the simulations with the proposed multiscale mass scaling approach. The black line represents ideal speedup. As you can see, a very large number of degrees of freedom were able to be solved and scalability was very similar. These simulations were carried out using the RedSky supercomputer (500 TeraFlops) at Sandia National Laboratories.

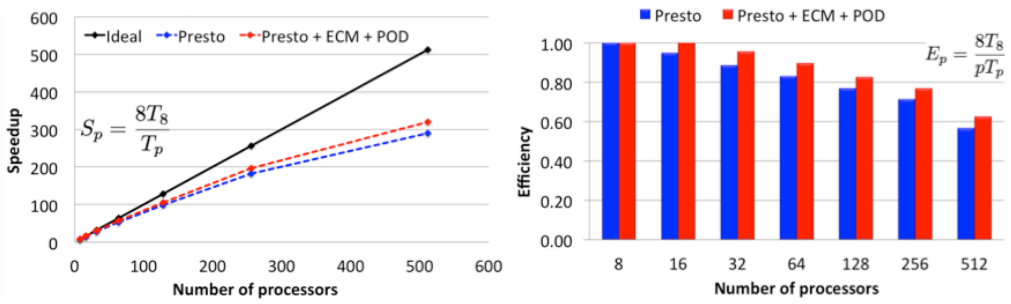


(a) (b)

Figure 2.12: a) Speedup and b) Efficiency results for 346,464 element problem in Example 1

2.5 Summary and conclusion

In this work, we developed a selective mass scaling strategy for accelerating explicit dynamic simulations in solid mechanics using a multiscale decompo-



(a) (b)

Figure 2.13: a) Speedup and b) Efficiency results for 2,771,712 element problem in Example 1

sition scheme. Proper orthogonal decomposition was used to efficiently build the coarse space of the decomposition. The two examples studied in this work showed that the proposed strategy can yield significant savings in computing time by increasing the stable time step, while maintaining reasonable levels of accuracy. However, finite element discretization, explicit time stepping, mass scaling, and proper orthogonal decomposition can all serve as sources of error in this method. The proper quantification of these sources of error is of paramount importance for devising sound strategies for selecting time steps, dimensionality of the coarse space, and the time spans at which snapshots are obtained. The ultimate goal is to find a suitable balance between accuracy and performance.

CHAPTER 3

EQUATION-FREE METHOD FOR EXPLICIT TIME INTEGRATION IN SOLID MECHANICS USING POD

In Chapters (1) and (2), we have discussed the need of developing fast and accurate computational methods to address problems in which the fine scale details have a significant influence on the overall coarse scale behavior of a problem. Our results, presented in the previous chapter, showed that indeed significant reductions in computational costs can be achieved while maintaining accuracy. However, we are aware that these multiscale problems have been tackled in a variety other ways; the equation-free method being one of them [41]. In this work, we develop an extension of this equation-free approach for the solution of solid mechanics problems.

Here, the equation-free approach and the mass scaling method proposed in (2.2.1), in essence, attempt to reduce the computational effort for obtaining accurate solutions to our initial boundary value problem shown in Eq. (2.10). However the manner on which they accomplish this is fundamentally different, and will be described in this chapter.

In general, the equation-free method carries out bursts of fine scale numerical simulations, and later uses the results to estimate important quantities in the coarse scales. The term "equation-free" stems from the fact that the equations that govern the evolution of the coarse scales are not defined in closed-form, but rather *observed* and *estimated* using these bursts of fine scale simulations. This approach was first documented by Theodoropouls *et. al.* [41], and used in the context of bifurcation analysis of a reaction-diffusion model. Since then, it has found to be useful in a wide range of different areas like molecular

dynamics, fluid flow, population dynamics, optimization, dynamical systems, weather simulations, among others [21, 23, 28, 22, 42, 3, 11]. In addition, the equation-free approach has been used in combination with other techniques like gap-tooth schemes [16, 32, 34], patch dynamics [22, 21, 23, 35, 3], and proper orthogonal decomposition (POD) [37, 28, 11, 42], to assist in the observation and evolution of the coarse scale dynamics based on the fine scale simulations. In this regard, this work will focus on using POD.

In Chapter (2), we described POD as a technique to obtain an optimal low-dimensional representation, or POD modes, of an ensemble of high dimensional data. This is one of the reasons that make POD modes a viable option for building a coarse space representation of our fine scale computational data. Traditionally, POD has been widely used for model reduction[1, 26, 5, 6, 33, 29, 14, 20], where a lower dimensional system is produced by the Galerkin projection of the governing equations on the POD modes. However, as pointed out by Sirisup *et. al.* [37], these reduced systems can lead to spurious asymptotic states, specially after long term integration. In order to alleviate these concerns, an equation-free/POD approach was developed [37, 28, 11, 42], for which a Galerkin projection does not take place, and the evolution of the coarse scale dynamics are *estimated*, rather than solved in closed-form. Hence, this equation-free approach has also been referred to as a Galerkin-free method [37].

For the equation-free method presented herein, the few most dominant POD modes are used for getting accurate low dimensional representations of fine scale numerical simulations. These representations are subsequently used to observe the evolution of the coarse scales, which is accomplished by an equation-free projective integration of the time dependent POD coefficients. Conse-

quently, POD modes only need to be computed once throughout the simulation.

This equation-free approach has shown, in other applications, the capability to produce fast and accurate solutions. Additionally, consistency and accuracy analysis have been developed[37] in the literature. For these reasons, we saw the possibility of improving upon the accuracy of the multiscale mass scaling method described in Chapter (2). This serves as our motivation to extend this method to solid mechanics.

The content of this chapter is organized as follows. In Section (3.1), a detailed description of the equation-free approach for solid mechanics is presented, using proper orthogonal decomposition. Numerical results are shown in Section (3.2), using two linear elastic solid mechanics examples, and discussing our findings. Finally, Section (3.3) contains a summary of the chapter, and discusses the remaining open issues and future directions of this work.

3.1 Formulation

The equation-free method presented herein is designed to obtain fast and accurate solutions to solid mechanics problems in which slow scales dominate. In other words, we are seeking solutions to the semi-discrete momentum equations shown in Expression (2.18). That is,

$$[M]\{\ddot{u}\} = \{f\}. \tag{3.1}$$

where $[M]$ is the mass matrix, $\{\ddot{u}\}$ are the accelerations, and $\{f\} = \{f_{ext}\} - \{f_{int}\}$ represents the difference between external and internal force vectors.

Much like the proposed method in (2.2.1), we assume that a separation of scales exists, and a constructed set of POD modes can be used to represent the

coarse scales. As seen in Section (2.1.1), we can obtain, from high dimensional displacement data, a set of orthogonal basis (POD modes) that can represent these displacement fields as

$$\boldsymbol{u}(\boldsymbol{x},t) = \sum_{j=1}^{m} a_j(t)\boldsymbol{\phi}_j(\boldsymbol{x}) \tag{3.2}$$

where $\boldsymbol{\phi}_j(\boldsymbol{x})$ is the j^{th} POD mode and $a_j(t)$ is the time dependent POD coefficient. Further details and derivations regarding POD are documented in Chapter(2).

It is important to point out that once the POD basis $\{\boldsymbol{\phi}_j(\boldsymbol{x})\}$ is obtained, the POD coefficients can be recovered by

$$a_j(t) = \int_{\Omega} \boldsymbol{u}(\boldsymbol{x},t)\boldsymbol{\phi}_j(\boldsymbol{x})d\Omega \equiv (\boldsymbol{u}(\boldsymbol{x},t),\boldsymbol{\phi}_j(\boldsymbol{x})). \tag{3.3}$$

The operation described in Equation (3.3) is called a *restriction* to the coarse space, in which the POD coefficients are computed using a displacement function $\boldsymbol{u}(\boldsymbol{x},t)$, and POD basis $\boldsymbol{\phi}_j(\boldsymbol{x})$. Defining a restriction operator $\mathcal{R}$ we can rewrite Eq. (3.3) as

$$\boldsymbol{a}(t) = \mathcal{R}\boldsymbol{u}(\boldsymbol{x},t). \tag{3.4}$$

Similarly, we define a *lifting* operator $\mathcal{L}$ as

$$\boldsymbol{u}(\boldsymbol{x},t) = \mathcal{L}\boldsymbol{a}(t) \tag{3.5}$$

$$= \sum_{j} a_j(t)\boldsymbol{\phi}_j(\boldsymbol{x}). \tag{3.6}$$

that computes the displacement function $\boldsymbol{u}(\boldsymbol{x},t)$ from the POD coefficients $a_j(t)$ and POD basis $\boldsymbol{\phi}_j(\boldsymbol{x})$. Both restriction and lifting operations represent an important part of the formulation of the equation-free method.

3.1.1 Equation-free method for solid mechanics using POD

By now, we have seen that POD is a technique that can use information from the fine scales (displacements) to build an accurate representations of the coarse scales. The issue that still remains is how do we advance the dynamics of the coarse scale. For this purpose, the following second order differential equation will govern the evolution of the POD coefficients in time

$$\frac{d^2\boldsymbol{a}}{dt^2}(t) = \boldsymbol{f}\left(\frac{d\boldsymbol{a}}{dt}(t); \boldsymbol{a}(t); t\right), \tag{3.7}$$

where $\boldsymbol{f}$ is not known explicitly.

At this point, traditional practice for model reduction suggest that the RHS of Eq. (3.7) be derived in closed-form by the Galerkin projection of the solid mechanics equations (3.1) onto the POD basis. This Galerkin approach for model reduction has been widely studied and has enjoyed a good level of success. However, it has also been shown to fail when predicting long term dynamics of a system [15, 36]. Part of the reason for this is that the first few most dominant POD modes used in this Galerkin truncation are not able to fully capture the long-term dynamics of the problem, giving rise to increasingly larger modal expansions, which eventually defeats the purpose of using model reduction in the first place.

In order to circumvent these known issues, an equation-free/POD approach was proposed by Sirisup *et. al.* [37]. The term "equation-free" arises from the fact that $\boldsymbol{f}$ is not derived in closed-form, but estimated using fine scale simulations of our solid mechanics problem. The process of estimating the RHS of Eq. (3.7), and later numerically integrating it to resolve the evolution of the coarse scales is called "equation-free projective integration".

This equation-free/POD approach has been used in the past [37, 28, 11, 42] for fluid problems, and has typically described the evolution of the coarse scales with a first order ODE. This ODE is in turn integrated using forward Euler. For our proposed framework, we formulate this method for solid mechanics problems, and introduce a *central difference projective integration* approach for integrating the second order differential equation proposed in Eq. (3.7). This central difference approach makes the formulation consistent with the one used for our fine scale simulator.

In summary the equation-free approach used in this work consists of five main operations: *fine scale simulation*, *coarse scale construction*, *restriction*, *central difference projective integration* and *lifting*. The rest of this section will go into further detail for each of these steps.

Fine scale simulation

For the fine scale simulator in this approach, we have used the finite element method with explicit time integration in order to obtain fine scale numerical solutions for the initial boundary value problem in (2.10). However, it is worth noting that any stable, accurate and consistent numerical method designed to tackle Eqs. (2.10) can be used within the equation-free framework. More details on the discretization for this fine scale simulator can be found in Section (2.1.2).

Implicit and explicit methods for time integration of transient problems have been well described in the literature [4, 8, 19, 7]. Even though explicit methods are conditionally stable, a system of equations does not need to be solved at each time step, making them computationally efficient. However, it is worth noting

that they do require very small time steps to be taken for very fine mesh sizes. For this work we have focused on using explicit central difference integration to integrate Eq. (3.1) in time.

Setting the start time of the simulation at $t = 0$, the final time t_f will be reached by taking multiple time steps Δt^n, where n represents the current time during the simulation. At any time step n, the semidiscrete momentum equation in (3.1) is given by

$$[M]^n\{\ddot{u}\}^n = \{f\}^n. \tag{3.8}$$

The central difference method uses central difference formulas for the velocity and acceleration as

$$\{\dot{u}\}^{n+\frac{1}{2}} = \frac{\{u\}^{n+1} - \{u\}^n}{t^{n+1} - t^n}, \tag{3.9}$$

$$\{\ddot{u}\}^n = \frac{\{\dot{u}\}^{n+\frac{1}{2}} - \{\dot{u}\}^{n-\frac{1}{2}}}{t^{n+\frac{1}{2}} - t^{n-\frac{1}{2}}}, \tag{3.10}$$

where $n+\frac{1}{2}$ and $n-\frac{1}{2}$ represent the midpoints of the next and previous time steps respectively. Eqs. (3.9) and (3.10) can be rearranged to obtain

$$\{u\}^{n+1} = \{u\}^n + (t^{n+1} - t^n)\{\dot{u}\}^{n+\frac{1}{2}}, \tag{3.11}$$

$$\{\dot{u}\}^{n+\frac{1}{2}} = \{\dot{u}\}^{n-\frac{1}{2}} + (t^{n+\frac{1}{2}} - t^{n-\frac{1}{2}})\{\ddot{u}\}^n. \tag{3.12}$$

Combining Eqs. (3.11) and (3.12) with Eq. (3.8) now enables us to obtain solutions to the solid mechanics problem described in Eq.(2.10).

In summary the explicit central difference algorithm used in the fine scale simulator is given as follows.

1. At $t = 0$ or $t = t_l$, set initial conditions and initialize fine scale simulator
2. Compute initial forces $\{f\}^0$

3. Compute accelerations $\{\ddot{u}\}^n$ from Eq. (3.8)

4. While $t \leq t_{final}$

   (a) Update time

   (b) Compute nodal velocities at midpoints with Eq. (3.12)

   (c) Compute nodal displacements with Eq. (3.11)

   (d) Update forces $\{f\}^{n+1}$

   (e) Compute new acceleration $\{\ddot{u}\}^{n+1}$ using Eq. (3.8)

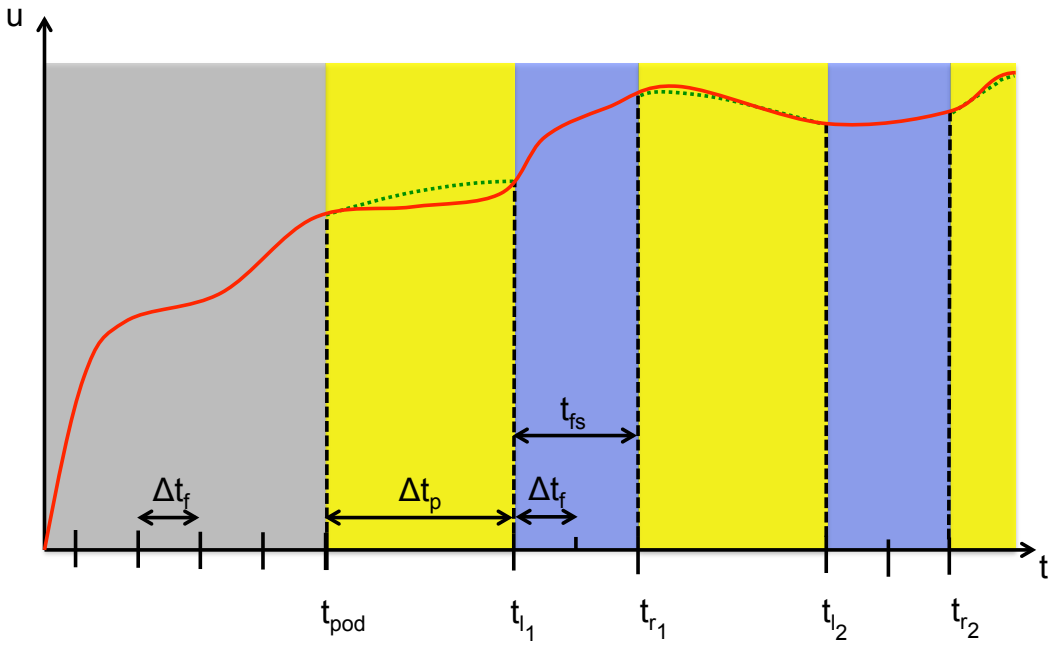


Figure 3.1: Sketch of the equation-free approach

Coarse scale construction

The next step in this method consists of computing the POD basis and coefficients. We have seen throughout this work that POD modes are good candidates

to represent the coarse space due to their optimal properties for representing an ensemble of high dimensional data. Additionally we are able to find the most dominant POD modes of the simulation, since they correspond to the highest eigenvalues of the correlation matrix.

In order to obtain data to build the POD modes, we need to start our simulation using the fine scale computation described in the previous section, while gathering snapshots of the displacement solution. During the fine scale computation, the ensemble of functions $\{\boldsymbol{u}_k\}_{k=1}^{n}$ is obtained by collecting displacement solutions at $t = n_s \Delta t_f$, where n_s is the number of fine scale time steps Δt_f taken between each snapshot. For example, if $n_s = 1$, displacement solutions are collected at each fine scale time step Δt_f. In Figure (3.1), $0 \leq t \leq t_{pod}$ represents where the initial fine scale computation takes place, and the displacements are collected. At time $t = t_{pod}$, the POD modes are constructed. More information about POD mode computation and construction of the coarse space can be found in Section (2.2.2).

Note that with this approach, POD modes are only computed once during the course of the simulation, at $t = t_{pod}$. The advancement of the coarse grained variables is permitted through the projective integration of the time-dependent POD coefficients.

The following step in the equation-free approach will be the restriction of the displacement field $\boldsymbol{u}(\boldsymbol{x}, t)$ in order to obtain the POD coefficients $\boldsymbol{a}(t)$. This operation will allow us to estimate the RHS of Eq. (3.7), which governs the evolution of the POD coefficients.

Restriction

The following step in this equation-free framework is to compute the POD coefficients $\boldsymbol{a}(t)$, through the restriction of the displacement field $\boldsymbol{u}(\boldsymbol{x}, t)$, using the POD modes $\boldsymbol{\phi}_j(\boldsymbol{x})$. A restriction step will be performed right after the POD modes are initially computed at time $t = t_{pod}$, and subsequently when a fine scale simulation ends at t_{r_m}, where m represents the number of restrictions to be taken; until the final time t_f is reached. The instants in time where the restriction steps take place ($t = t_{pod}$, and t_{r_m}) are sketched in Figure (3.1).

After obtaining the POD basis $\boldsymbol{\phi}_j(\boldsymbol{x})$, the POD coefficients are computed using Eq. (3.4) as

$$\boldsymbol{a}(t^n) = \left(\boldsymbol{u}(\boldsymbol{x}, t^n), \boldsymbol{\phi}_j(\boldsymbol{x})\right). \tag{3.13}$$

Part of the restriction step also encompasses the estimation of the first and second derivatives of the POD coefficients for Eq. (3.7) as

$$\dot{\boldsymbol{a}}(t) = \left(\dot{\boldsymbol{u}}(\boldsymbol{x}, t), \boldsymbol{\phi}_j(\boldsymbol{x})\right), \text{ and} \tag{3.14}$$

$$\ddot{\boldsymbol{a}}(t) = \left(\ddot{\boldsymbol{u}}(\boldsymbol{x}, t), \boldsymbol{\phi}_j(\boldsymbol{x})\right), \tag{3.15}$$

respectively. This step will allow us to perform the central difference projective integration described in the next section. It is worth noting that these estimations are easily obtained using the accelerations $\ddot{\boldsymbol{u}}(\boldsymbol{x}, t)$ and velocities $\dot{\boldsymbol{u}}(\boldsymbol{x}, t)$ computed with our fine scale explicit dynamics simulator.

Central difference projective integration

Using Eqs. (3.13), (3.14), and (3.15); we can proceed to integrate Eq. (3.7) using a central difference projective integration. It is worth noting that any traditional

solver can be used for this purpose. In this work, using central differences for the integration in the coarse scales is consistent with the integration scheme used for the fine scale simulator. It is important to point out that previous work [37, 28, 11, 42] done with equation-free/POD methods have postulated the evolution of the POD coefficients in (3.7) with a first order ODE, and have integrated it using forward Euler. In our work, we introduce a *central difference projective integration* approach for integrating the second order differential equation proposed in Eq. (3.7).

One of the characteristics of central difference algorithms is that the velocities are computed at the midpoints of the time steps used in the simulation. Therefore, at any time $t = t_n$, the first derivative estimates in Eq. (3.14) is computed as

$$\dot{\boldsymbol{a}}\left(t^{n-\frac{1}{2}}\right) = \left(\dot{\boldsymbol{u}}\left(\boldsymbol{x}, t^{n-\frac{1}{2}}\right), \boldsymbol{\phi}_j(\boldsymbol{x})\right), \tag{3.16}$$

while the restriction in Eq. (3.13) and the second derivative estimate from Eq. (3.15) are obtained as

$$\boldsymbol{a}\left(t^n\right) = \left(\boldsymbol{u}\left(\boldsymbol{x}, t^n\right), \boldsymbol{\phi}_j(\boldsymbol{x})\right), \tag{3.17}$$

$$\ddot{\boldsymbol{a}}\left(t^n\right) = \left(\ddot{\boldsymbol{u}}\left(\boldsymbol{x}, t^n\right), \boldsymbol{\phi}_j(\boldsymbol{x})\right). \tag{3.18}$$

Using Eqs. (3.16), (3.17), and (3.18) we can now use central difference projective integration to obtain

$$\dot{\boldsymbol{a}}\left(t^{n+\frac{1}{2}}\right) = \dot{\boldsymbol{a}}\left(t^{n-\frac{1}{2}}\right) + \frac{1}{2}\left(\Delta t_f + \Delta t_p\right)\ddot{\boldsymbol{a}}\left(t^n\right) \tag{3.19}$$

$$\boldsymbol{a}\left(t^{n+1}\right) = \boldsymbol{a}\left(t^n\right) + \left(\Delta t_p\right)\dot{\boldsymbol{a}}\left(t^{n+1}\right), \tag{3.20}$$

where Δt_p is the coarse projective time step; and Δt_f represents the time step of the fine scale computations that is often equal to the critical element time step

Δt_c. The areas $t_{pod} \leq t \leq t_l$ and $t_r \leq t \leq t_l$ from Figure (3.1) represent where the central difference projective integration is taking place.

Lifting

After the central difference projection of displacements and velocities at a coarser time step Δt_p, the next step consist of lifting to the fine scales these coarse grained quantities computed in (3.19) and (3.20), using Eq. (3.5)

$$\dot{\boldsymbol{u}}(\boldsymbol{x}, t^{n+\frac{1}{2}}) = \sum_j \dot{a}_j(t^{n+\frac{1}{2}})\boldsymbol{\phi}_j(\boldsymbol{x}). \tag{3.21}$$

$$\boldsymbol{u}(\boldsymbol{x}, t^{n+1}) = \sum_j a_j(t^{n+1})\boldsymbol{\phi}_j(\boldsymbol{x}). \tag{3.22}$$

Expressions (3.21) and (3.22) become now the starting point of a new fine scale simulation.

3.1.2 Equation-free/POD algorithm flow and implementation notes

The main steps taken during this equation-free simulation will now be summarized in this section, as we outline the flow of the algorithm.

1. For $0 \leq t \leq t_{pod}$, perform a *fine scale simulation* using $\Delta t_f \leq \Delta t_c$, and gather snapshots of the nodal displacements $\boldsymbol{u}(\boldsymbol{x}, t^n)$ at a user defined interval n_s.
2. At $t = t_{pod}$, *construct the coarse space* by computing POD modes using the gathered snapshots and Eq. (2.9).
3. While $t \leq t_f$

(a) Perform *restriction* of displacement vector using Eq. (3.17). Estimate velocities and accelerations with Eqs. (3.16) and (3.18).

(b) Set new time step $\Delta t_p \geq \Delta t_f$

(c) Execute *central difference projective integration* using Eqs. (3.19) and (3.20).

(d) Through *lifting*, take these coarse grained integrated quantities and compute new velocities and displacements as observed in Eqs. (3.21) and (3.22)

(e) Use lifted velocities and displacements as initial conditions and perform *fine scale computations* for a period of time $t = t_{fs}$.

Due to nature of this algorithm, where multiple projections of the corse scales are taking place, the implementation will be limited to problems with materials with no history dependence. Future work could be focused in devising effective strategies to project, in a consistent manner, relevant internal variables of history dependent materials.

This algorithm was implemented in the SIERRA Mechanics code developed at Sandia National Laboratories. SIERRA consists of a suite of analysis modules that posses multi-physics capabilities to tackle many of Sandia's scientific applications. All of the work was done in the solid mechanics module, SIERRA/SM[40], which is a 3D implicit and explicit transient dynamics code that analyzes structures and solids. As noted in Section (2.4), all of the capabilities to perform Proper Orthogonal Decomposition were implemented. For this algorithm, we extended these capabilities to handle the requirements set forth by our equation-free formulation, like central difference projective integration, POD coefficient computation, restriction and lifting from coarse to fine

scales. Also, the coupling between the existing explicit dynamics module with our equation-free approach was implemented for this method. The environment created in the implementation of this algorithm allows the user to easily modify important variables for this approach as coarse integration time steps, and frequency of projective time integration procedures.

3.1.3 Remarks on consistency and accuracy

It is important to point out that the described equation-free/POD approach has shown to be consistent and accurate. Sirisup *et. al.* [37] demonstrated consistency and accuracy of this approach when a first-order Euler projective integrator was used. Additionally, it is remarked that these properties will be conserved for any integration scheme (higher order included), as long as the scheme itself has proven to be consistent and accurate. This holds true for the central difference projective integration approach used in our work.

Specifically for consistency, the equation-free/POD method shows that as the time step and maximum mesh spacing approach zero, and the number of terms in the POD expansion moves towards infinity, the true solution of the problem will be recovered. In terms of accuracy, the error in this equation-free/POD approach was obtained as

$$\epsilon \sim \left(\delta t^{p}, h^{q}, \delta t^{J_f - 1}, \Delta t_c^{J_c}, \frac{K^{-\gamma}}{\Delta t} \right), \tag{3.23}$$

where we can see the dependance on time step size (δt^p) and mesh spacing (h^q) of the fine scale solver, the order of approximation of Eq. (3.7) $\left(\delta t^{J_f - 1}\right)$, the order of projective integration $\left(\Delta t_c^{J_c}\right)$, and the number of POD modes $\left(\frac{K^{-\gamma}}{\Delta t}\right)$. As the errors in the fine scale solvers decrease, the POD error and error in the order

of projective integration will dominate. It is important to note that the stability analysis for this method still remains an open research question, and should be addressed in future work.

3.2 Numerical Results

The numerical results presented in this section illustrate some of the advantages and disadvantages of using the proposed equation-free approach. Similar to the numerical results presented in Section (2.3), the concept of computational performance will be addressed, as well as accuracy. We will also discuss comparisons with conventional central difference algorithms and the multiscale mass scaling method described in Chapter (2).

3.2.1 Beam bending

This first example consists of a 3D steel cantilever beam, considered entirely linear elastic. Fixed displacement boundary conditions were only specified on one surface of the beam. A transient pressure load was applied on the top surface of the beam, shown in Figure (3.2). The load amplitude was increased linearly from 0 to 2 psi for a total simulation time of 5×10^{-3} s. The density and Young's modulus of the material were taken as 7.32×10^{-4} lbf $\cdot$ sec^2/in^4 and 30×10^6 psi, respectively. A relatively small mesh of 80 8-node hexahedral elements was used, which required a critical time step of 1.158×10^{-6} s.

For the initial fine scale simulation, a total of 21 snapshots were taken, and POD modes were computed 5×10^{-4} s into the simulation (10% of the total time).

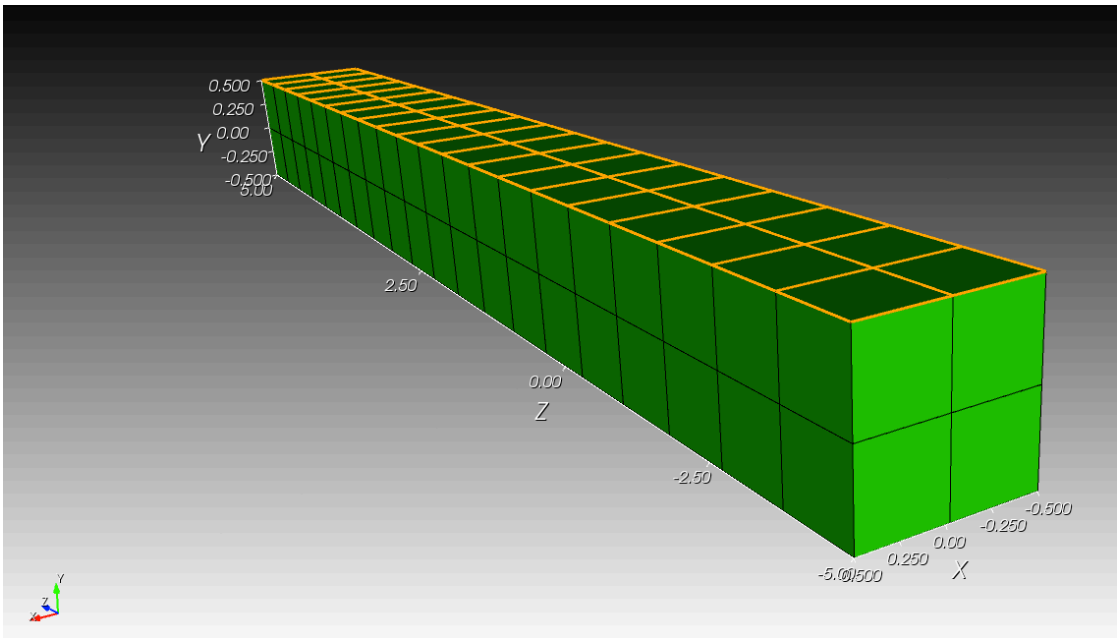


Figure 3.2: Beam mesh for equation-free example 1

The 3 most dominant POD modes were used in this problem, which accounted for $k(3) > 99.99\%$ from Eq. (2.38). Also, a projective integration step occurred every 10 critical time steps of fine scale simulation.

Figure (3.3) shows the displacement of the beam in the Y-direction of one of the nodes attached to the free end. For this plot, the size of the projective integration time step ranged from 0% to 50% larger than the critical element time step. This figure shows that, in this range, the solution approaches the original explicit dynamics dynamics solution as the time step approaches the critical one. However, simulations with projective time steps larger than 50% proved to be increasingly unstable. Further testing also found that reducing the number of projective integration time steps throughout the simulation alleviated some of these instabilities. This is expected, since reducing the number of projective integrations increases number of fine scale time steps that have to be

taken. This obviously affects performance, but eventually leads to the reduction of dominant errors produced by the projective integrator.

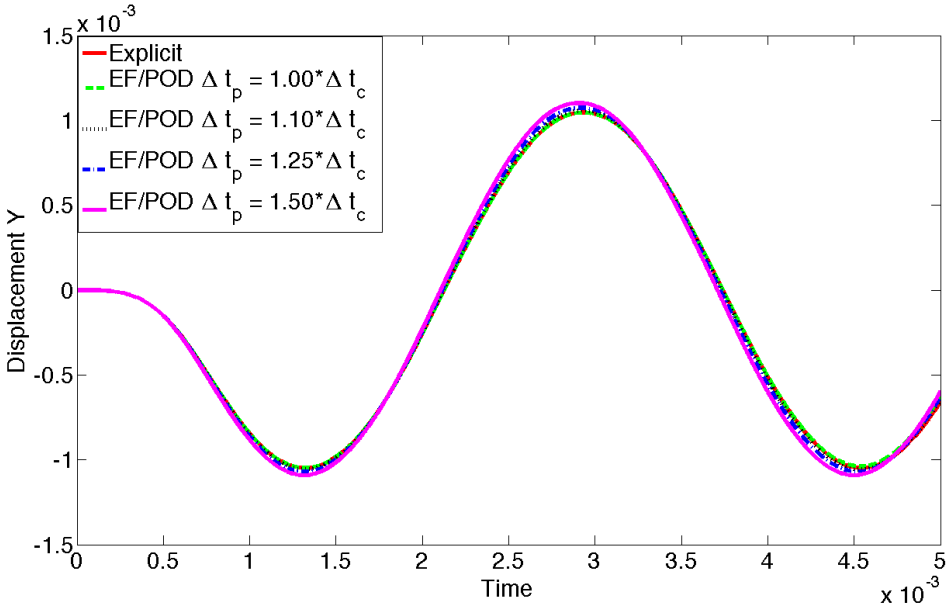


Figure 3.3: Displacement Y at free tip of beam. Comparison between equation-free runs at different time steps

Comparisons were also made with the multiscale mass scaling method presented in Chapter (1). For this problem, results were more accurate with the multiscale mass scaling method as opposed to the equation-free one. Figure (3.4) shows the comparison between the mass scaling approach and equation-free when using time steps 50% larger than the critical element one for both coarse time stepping and projective integration respectively. When compared to traditional explicit dynamics, the relative error for the maximum displacement *Y*of the beam at the free end was 5.36% for equation-free, and 0.31% for the mass scaling approach.

In Figure (3.5) we illustrate how the solution can become unstable when a large projective integration time step is used. This plot compares the solution

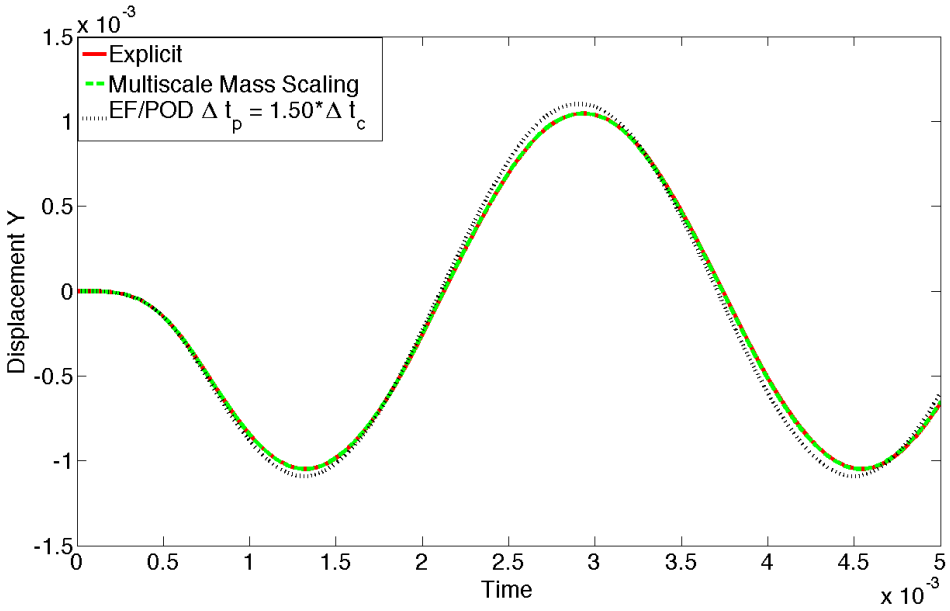


Figure 3.4: Displacement Y at free tip of beam. Comparison between explicit dynamics, equation-free, and mass scaling approach at 50% larger time step.

of traditional explicit dynamics with one using the equation free approach at projective integration time steps 65% larger than the critical one, and where the fine scale simulations ran for 10 critical time steps.

Numerous cases of this problem were run, with varying projective integration time steps. These cases were studied, then compared to results obtained from a central difference explicit dynamics algorithm using the critical element-based time step Δt_c. The results show that when more fine scales simulations are used, and when the size of the projective integration time step is decreased, we obtain better agreement with the traditional fine scale explicit dynamics solution. However, it is worth noting that, for this problem, a 50% increase in the critical element time step produced accurate results. This is a noticeable increase that leads to overall performance improvements.

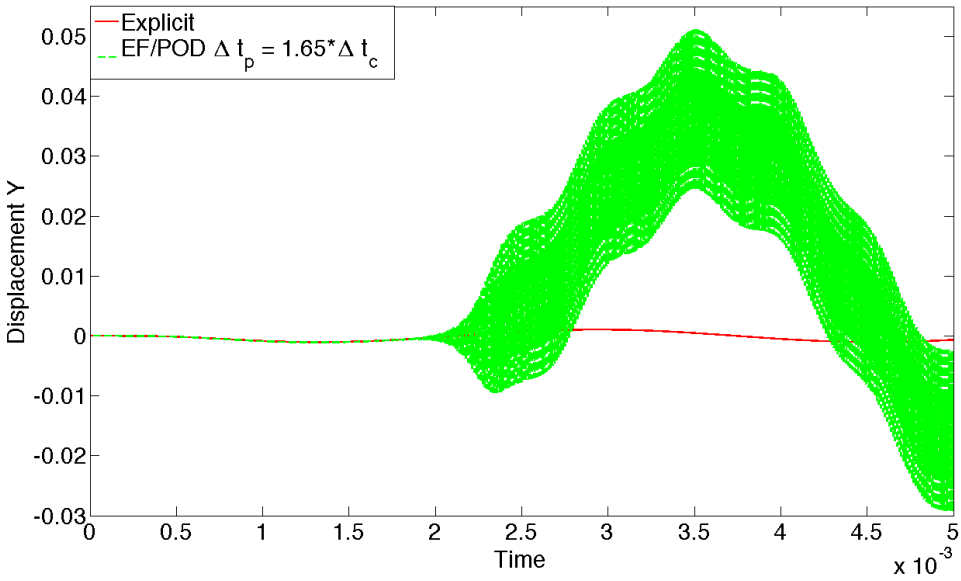


Figure 3.5: Displacement Y at free tip of beam. Comparison between explicit dynamics, and equation-free/PO at 65% larger time step.

3.2.2 Linear elastic cylinder with legs

The purpose of this example is to tackle a more challenging problem using this equation-free approach. Also, we want to compare it to the solutions obtained in Section (2.3.1) using the multiscale mass scaling method. In this example we have a linear elastic hollow steel cylinder with legs. The domain is the same as the numerical example tackled in Section (2.3.1). A transient pressure load was applied on a small square area of the side of the cylinder as shown in Figure (2.2). The bottom surfaces of the legs have fixed displacement boundary conditions. The load amplitude was increased linearly from 0 to 10^5 psi for a total simulation time of 5×10^{-4} s. The density and Young's modulus of the material were taken as 7.32×10^{-4} lbf $\cdot$ sec^2/in^4 and 30×10^6 psi, respectively. The mesh used for this problem, shown in (2.2) consisted of 43,308 8-node hexahedral elements,

and the corresponding element-based critical time step was 1.23×10^{-8} s.

Here, 20 snapshots were collected from the initial fine scale simulator. The construction of the coarse space started at $t_{pod} = 1 \times 10^{-4}$ s, accounting for 20% of the total simulation time. The first 4 POD modes where used, which accounted for $k(4) > 99.99\%$ from Eq. (2.38). Different cases were run, where projective integration time steps occurred every 25, 50 and 100 times the size of the critical time step. The purpose of this was to study the effect of increasing the number of fine scale simulations on the solution. Also, results obtained using this equation-free approach were compared to ones that use the same time step for the multiscale mass scaling method presented in Section (2.3.1).

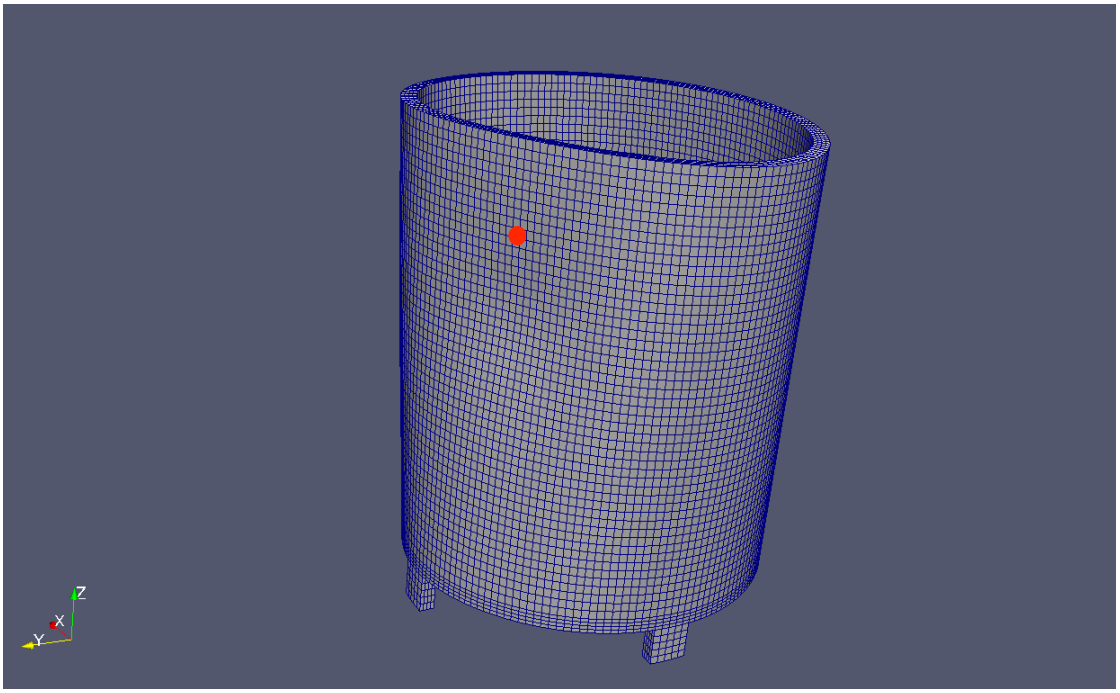


Figure 3.6: Selected node for the equation-free/POD plots in this section

The relative error described in Eq.(2.37), was computed using all of the time steps and nodes of the domain. When using the equation-free approach, the

error was as 8.53%, 7.59%, and 7.11% for simulations where the fine scale simulator lasted for $25\Delta t_c$, $50\Delta t_c$, and $100\Delta t_c$, respectively. This shows that, as the number of fine scale simulations is increased, the errors will decrease, but performance would be affected. For the multiscale mass scaling method this error was 1.17%

In Figure (3.7) we are plotting the displacement magnitude at a point located just above the nodes where the load is applied (see Figure (2.2)). The node used in these plots is shown in Figure (3.6). This figure shows the comparison between the equation-free/POD simulations using time steps 25%, 50%, 75%, and 100% larger than the critical one. Here, the solutions slightly improve as the size of the projective time step approaches the critical element one. All of the simulations plotted in this figure ran the fine scale simulator for 100 critical time steps.

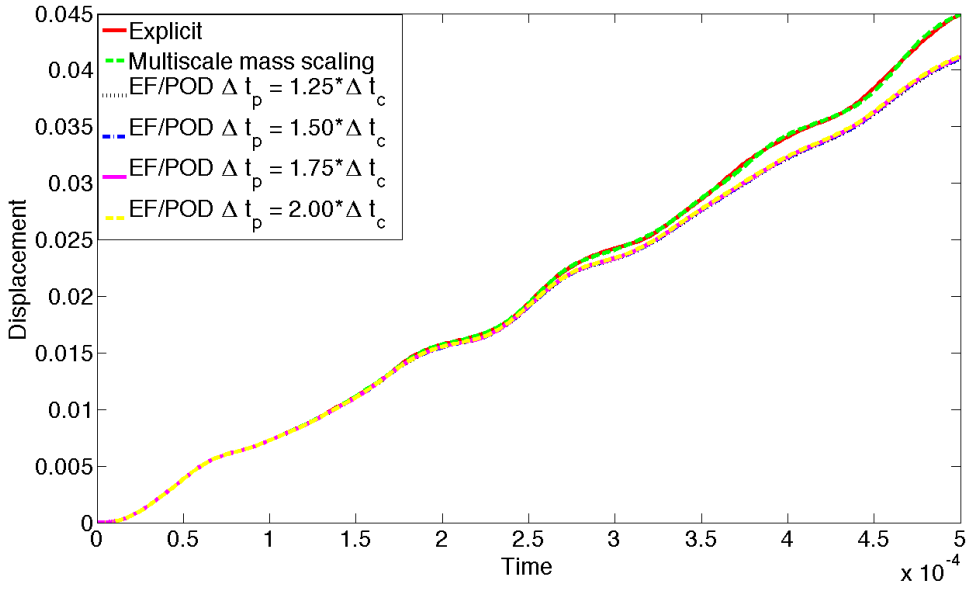


Figure 3.7: Displacement magnitude at a point at different projective integration time steps

In Figure (3.8) we are plotting the displacement magnitude at the same point seen in Figure (3.6). Here, we compare simulations of the equation-free approach where the fine scale simulator lasts for 25, 50, and 100 times the size of the critical time step. Also, the comparison is made to the results produced by the mass scaling method described in Chapter (2). In this figure, all of the simulations used a coarse time step 100% larger than the critical element one, except for the reference solution in explicit dynamics. This plot shows that with the multiscale mass scaling method we obtained better agreement with the solution than the equation-free approach.

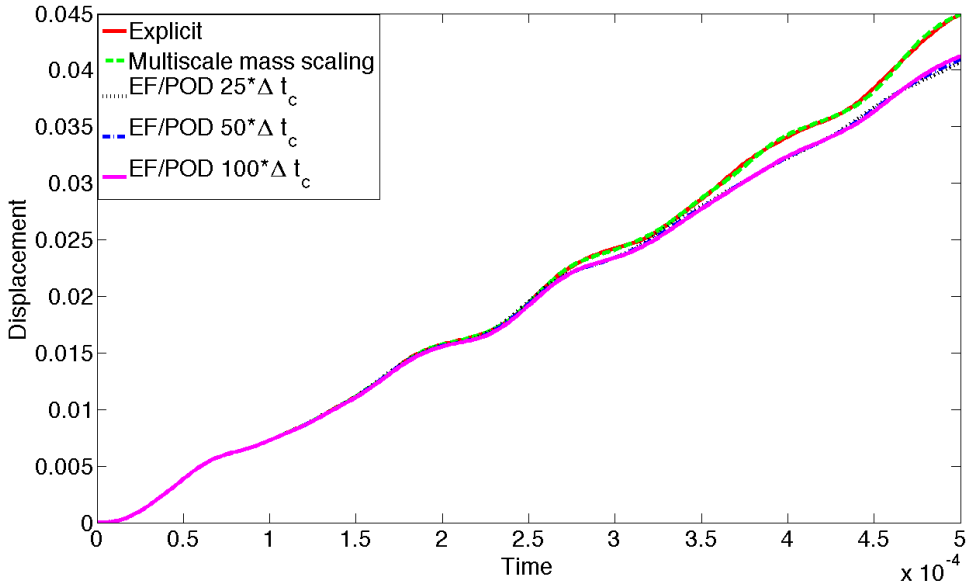


Figure 3.8: Displacement magnitude at a point at different sizes of fine scale integration. Projective integration time step 100% larger than the critical one.

3.3 Summary and Conclusions

In this chapter, we developed a computational framework, centered in the equation-free approach, for the solution of multiscale solid mechanics problems using Proper Orthogonal Decomposition. This approach was used in order to circumvent some of the drawbacks with traditional Galerkin projections for transient dynamics problems. For this approach, POD was used to build the coarse scales by using displacement data from the fine scale explicit dynamics simulator. We introduce central difference projective integration to estimate and advance the dynamics of these coarse scales. The examples show that as the number of fine scale simulations increase, as expected, the error decreases. This is also the case when the projective integration time step approaches the critical one. Comparisons of this equation-free/POD approach shows that the multiscale mass scaling method proposed in Chapter (2) displayed better performance, in general. However, the consistency and accuracy properties of this method and the potential of performance improvements for long term dynamics problems still make this method attractive. Furthermore, the method has enjoyed success in other engineering applications. Our limited evidence did expose some faults, but further investigation is worth pursuing. Future directions for this work can focus on improving the overall accuracy and performance of the method by studying the selection of the fine scale simulators and the order of the projective integration. Eventually, the ultimate goal is to save computational time while obtaining accurate solutions for our solid mechanics problems.

CHAPTER 4

CONCLUSION

In this work, I have focused on tackling multiscale solid mechanics problems, through the development and implementation of novel algorithms. All of the code development described in this work took place in Sandia National Laboratories' SIERRA mechanics suite, specifically its Solid Mechanics (SM) module. We were able to develop and integrate our algorithms to existing code and libraries, while maintaining usability, scalability and performance standards for which SIERRA was designed. This process was both very challenging and rewarding.

Future directions of this work can readily be conceived. For both methods described here, it is important to further investigate how all of the sources of error relate to each other, and affect our numerical solutions. Also, new ideas that can provide further guidance in choosing an appropriate amount of snapshots and POD modes "on the fly", during a simulation, and that can guarantee accuracy, while still delivering performance improvements. For the multiscale method presented in Chapter (2), we can explore how different types of mass scaling can affect the overall solution of the problem, and try to devise a way to optimally scale the mass. A natural extension of the equation-free approach in solid mechanics is the capability of tackling nonlinear problems. Additionally, both methods could be explored for broader applications involving multiphysic phenomena.

One of the main contributions of this thesis was to effectively integrate and use the described multiscale mass scaling scheme, with proper orthogonal decomposition. We extensively studied the effect of various factors in the overall

performance and accuracy of the method. In the end, we were able to observe accuracy, while obtaining tremendous performance improvements in computational time. Another contribution represents the application of an equation-free/POD approach to solid mechanics, that used a central difference projective integrator to estimate important quantities in the coarse scales.

APPENDIX A

NODAL BASED TIME STEP DERIVATION

The expression shown in (2.31) for the scaling factor α_i was derived from the nodal-based stable time step estimate developed by Heinstein *et. al.* [17]. The derivations of the stable time step reported by these authors are described herein for the sake of completeness. Heinstein *et. al.* arrived at a time step stability criterion expressed in terms of nodal mass and stiffness given as

$$\Delta t_c \leq \frac{2}{\sqrt{\frac{\hat{K}^i}{M^i}}\Big|_{\text{max over i}}}, \tag{A.1}$$

where M^i is the lumped mass at node i, and $\hat{K}^i$ is the assembly of the maximum element modal stiffness, given by $\sum_{e \subset e^i} k^e_{\max}$, where e^i is the set of elements that are connected to node i. To derive this nodal-based time step we start with the stable time step reported by Krieg and Key in [25] as

$$\Delta t_c \leq \frac{2}{\sqrt{\lambda_{\max}}}, \tag{A.2}$$

where λ_{max} represents the maximum global eigenvalue of the system. In order to arrive at a nodal-based form of the maximum eigenvalue estimate given above, we consider the generalized eigenvalue problem

$$([K] - \lambda[M])\{\overline{u}\} = 0, \tag{A.3}$$

where $[K]$ is the stiffness matrix, and $[M]$ is the lumped mass matrix. Let $\lambda_{\max}$ denote the largest eigenvalue and $\{\overline{u}\}_{\max}$ the corresponding eigenvector. Then, the Rayleigh quotient for the maximum eigenvalue $\lambda_{\max}$ is given as

$$\lambda_{\max} = \frac{\{\overline{u}\}^T_{\max}[K]\{\overline{u}\}_{\max}}{\{\overline{u}\}^T_{\max}[M]\{\overline{u}\}_{\max}}, \tag{A.4}$$

where the numerator, in a finite element framework, can be expressed as

$$\{\overline{u}\}^T_{\max}[K]\{\overline{u}\}_{\max} = \sum_{e=1}^{N_e}(\{\overline{u}\}^e_{\max})^T[K]^e\{\overline{u}\}^e_{\max}. \tag{A.5}$$

In the above equation, N_e corresponds to the number of elements in a finite element mesh. We also consider an eigenvalue problem for the element stiffness matrix as

$$[K]^e\{\phi\}^e = \beta^e\{\phi\}^e, \tag{A.6}$$

where $\{\phi\}^e$ and β^e are, respectively, an eigenvector and the corresponding eigenvalue of the element stiffness matrix. Since the element eigenvectors span their space, we can show that for any vector $\{v\}^e$ in this space, the following inequality holds.

$$(\{v\}^e)^T[K]^e\{v\}^e \leq \beta^e_{\max}\|\{v\}^e\|^2, \tag{A.7}$$

where $\beta^e_{\max}$ is the maximum eigenvalue of the element stiffness matrix. Now, let's define a diagonal element stiffness matrix $[\hat{K}]^e$ as

$$[\hat{K}]^e := \beta^e_{\max}[I]^e. \tag{A.8}$$

Then, we can we can define a global diagonal stiffness matrix $[\hat{K}]$, which is assembled from the element matrices $[\hat{K}]^e$. Using Equations (A.7) and (A.8), we can obtain

$$\sum_{e=1}^{N_e}(\{\overline{u}\}^e_{\max})^T[K]^e(\{\overline{u}\}^e_{\max}) \leq \sum_{e=1}^{N_e}(\{\overline{u}\}^e_{\max})^T[\hat{K}]^e(\{\overline{u}\}^e_{\max}) \tag{A.9}$$

Now, defining $\hat{\lambda}_{max}$ as

$$\hat{\lambda}_{\max} = \frac{\{\overline{u}\}^T_{\max}[\hat{K}]\{\overline{u}\}_{\max}}{\{\overline{u}\}^T_{\max}[M]\{\overline{u}\}_{\max}}, \tag{A.10}$$

and using Eqs. (A.4) and (A.9), we obtain

$$\lambda_{\max} \leq \hat{\lambda}_{\max}. \tag{A.11}$$

Computing $\hat{\lambda}_{\max}$ is not trivial in most realistic problems. Therefore, instead of computing the exact value for $\hat{\lambda}_{\max}$, an upper bound can be obtained. To this end, we start by defining a ratio of nodal stiffness and mass for Node i as

$$\hat{\gamma}^i := \frac{\hat{K}^i}{M^i}. \tag{A.12}$$

We assume that these ratios are ordered such that $\hat{\gamma}^m \geq \hat{\gamma}^{m-1} \geq ... \geq \hat{\gamma}^i$ for integers $m > i$. The latter ordering implies

$$\hat{\gamma}^m = \left.\frac{K^i}{M^i}\right|_{\text{max over i}}. \tag{A.13}$$

Since $\hat{K}^i$ and M^i are diagonal, Eq. (A.10) can be rewritten as

$$\hat{\lambda}_{\max} = \frac{\sum_i (\overline{u}^i_{\max})^2 \hat{K}^i}{\sum_i (\overline{u}^i_{\max})^2 M^i}. \tag{A.14}$$

Using Equations (A.12), (A.10), and after some simple algebraic manipulations, we get

$$\hat{\lambda}_{\max} = \hat{\gamma}^m \left\{ \frac{1 + \frac{(\overline{u}^{m-1}_{\max})^2 M^{m-1}}{(\overline{u}^{m-1}_{\max})^2 M^m} \left(\frac{\hat{\gamma}^{m-1}}{\hat{\gamma}^m}\right) + ...}{1 + \frac{(\overline{u}^{m-1}_{\max})^2 M^{m-1}}{(\overline{u}^{m-1}_{\max})^2 M^m} + ...} \right\}. \tag{A.15}$$

Notice that $\frac{\hat{\gamma}^{m-i}}{\hat{\gamma}^m} \leq 1$. Hence, it follows that

$$\begin{aligned} \hat{\lambda}_{\max} &\leq \hat{\gamma}^m \\ &\leq \left.\frac{K^i}{M^i}\right|_{\text{max over i}} \end{aligned} \tag{A.16}$$

Finally, we can use the inequality in (A.16) with Eq. (A.2) to obtain the final expression for the nodal-based stable time step as

$$\Delta t_c \leq \frac{2}{\left.\sqrt{\frac{\hat{K}^i}{M^i}}\right|_{\text{max over i}}}. \tag{A.17}$$

BIBLIOGRAPHY

[1] W. Aquino. An object-oriented framework for reduced-order models using proper orthogonal decomposition (POD). *Comp. Meth. Appl. Mech. Eng.*, 196(41-44):4375–4390, 2007.

[2] W. Aquino, J. C. Brigham, C. J. Earls, and N. Sukumar. Generalized finite element method using proper orthogonal decomposition. *Int. J. Num. Meth. Eng.*, 79:887–906, 2009.

[3] A. Armaou, I. G. Kevrekidis, and C. Theodoropoulos. The gaptooth scheme, patch dynamics and equation-free controller design for distributed complex/multiscale processes. In *American Control Conference, 2004. Proceedings of the 2004*, volume 1, pages 926–932. IEEE, 2004.

[4] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. Wiley, West Sussex, England, 2000.

[5] G. Berkooz, P. Holmes, and J. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Ann. Rev. Fluid Mech.*, 25:539–575, 1993.

[6] K. Carlberg, C. Bou-Mosleh, and C. Farhad. Efficient nonlinear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations. *Int. J. Num. Meth. Eng.*, 86:155–181, 2011.

[7] K. J. Cook. *Finite Element Procedures*. Prentice-Hall, 1996.

[8] R. Cook, D. Malkus, M. Plesha, and R. Witt. *Concepts and Applications of Finite Element Analysis*. Wiley (4th Edition), New York, 2002.

[9] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics (translation of 1928 german original). *IBM Journal*, 11:215–234, 1976.

[10] H. C. Edwards and J. R. Stewart. Sierra: A software environment for developing complex multi-physics applications. In *First MIT Conference on Computational Fluid and Solid Mechanics*, pages 1147–1150, Amsterdam, 2001. Elsiver.

[11] V. Esfahanian and Khosro Ashrafi. Equation-free/galerkin-free reduced-order modeling of the shallow water equations based on proper orthogonal decomposition. *Journal of fluids engineering*, 131(7):C63757, 2009.

[12] D. P. Flanagan and T. Belytschko. Simultaneous relaxation in structural dynamics. *Journal of the Engineering Mechanics Division, ASCE*, 107:1039–1055, 1981.

[13] D. P. Flanagan and T. Belytschko. Eigenvalues and stable time steps for the uniform strain hexahedron and quadrilateral. *ASME Journal of Applied Mechanics*, 51:35–40, 1984.

[14] F. Galland, A. Gravouil, E. Malvesin, and M. Rochette. A global model reduction approach for 3D fatigue crack growth with confined plasticity. *Comp. Meth. Appl. Mech. Eng.*, 200(5-8):699–716, 2011.

[15] C.W. Gear and I. G. Kevrekidis. Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum. *SIAM J. Sci. Comput.*, 24:1091–1106, 2003.

[16] C.W. Gear, J. Li, and I. G. Kevrekidis. The gap-tooth method in particle simulations. *Phys. Lett. A*, 316:190–195, 2003.

[17] M. W. Heinstein, F. J. Mello, and C. R. Dohrmann. A nodal-based stable time step predictor for transient dynamics with explicit time integration. *ASME Piping Div Publ PVP*, 343:225–229, 1996.

[18] P. Holmes, J. Lumley, and G. Berkooz. *Turbulence, Coherent Structures, Dynamical Systems, and Symmetry.* Cambridge University Press, Cambridge, 1996.

[19] T. J. R. Hughes. *The Finite Element Method - Linear Static and Dynamic Finite Element Analysis*. Dover, New York, 2000.

[20] G. Kerschen, J. C. Golinval, A. Vakakis, and L. Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: An overview. 41(1-3):147169, 2005.

[21] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodoropoulos. Equation-free coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Commun. Math. Sci.*, 1(4):715–762, 2003.

[22] I. G. Kevrekidis and L. Mathelin. Equation-free multiscale computation: Algorithms and applications. *Annu. Rev. Phys. Chem.*, 60:321344, 2009.

[23] I.G. Kevrekidis, C. W. Gear, and G. Hummer. Equation-free: the computer-assisted analysis of complex, multiscale systems. *AIChE J.*, 50:1346–1354, 2004.

[24] J.R. Koteras and R. B. Lehoucq. Estimating the critical time-step in explicit dynamics using the Lanczos method. *Int. J. Num. Meth. Eng.*, 1:1–21, 2003.

[25] R.D. Krieg and S. W. Key. Transient shell response by numerical time integration. *Int. J. Num. Meth. Eng.*, 7:273–286, 1973.

[26] P. Krysl, S. Lall, and J.E. Marsden. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *Int. J. Num. Meth. Eng.*, 51:479–504, 2001.

[27] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research of the National Bureau of Standards*, 45:255–282, 1950.

[28] O. Le Maître and L. Mathelin. Equation-free model reduction for complex dynamical systems. *Int. J. Num. Meth. Fluids*, 63(2):163–184, 2009.

[29] A. Nouy. A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations. *Comp. Meth. Appl. Mech. Eng.*, 199(23-24):1603–1626, 2010.

[30] B. D. Reddy. *Introductory Functional Analysis with Applications to Boundary-value Problems and Finite Elements*. Springer, Berlin, 1997.

[31] R. D. Richtmeyer and K. W. Morton. *Difference Methods for Initial-Value Problems*. Interscience Publishers, 1967.

[32] A. J. Roberts and I. G. Kevrekidis. Higher order accuracy in the gap-tooth scheme for large-scale solutions using microscopic simulators. *ANZIAM J.*, 46:C63757, 2005.

[33] D. Ryckelynck and D. Missoum Benziane. Multi-level a priori hyper-reduction of mechanical models involving internal variables. *Comp. Meth. Appl. Mech. Eng.*, 199(17-20):1134–1142, 2010.

[34] G. Samaey, I. G. Kevrekidis, and D. Roose. Damping factors for the gap-tooth scheme. *Multiscale modelling and simulation*, pages 93–102, 2004.

[35] G. Samaey, D. Roose, and I. G. Kevrekidis. Combining the gap-tooth scheme with projective integration: patch dynamics. *Multiscale Methods in Science and Engineering*, pages 225–239, 2005.

[36] S. Sirisup and G. E. Karniadakis. A spectral viscosity method for correcting the long-term behavior of POD models. *Journal of Computational Physics*, 194(1):92–116, 2004.

[37] S. Sirisup, G. E. Karniadakis, D. Xiu, and I. G. Kevrekidis. Equation-free/Galerkin-free POD-assisted computation of incompressible flows. *Journal of Computational Physics*, 207:568–587, 2005.

[38] L. Sirovich. Turbulence and the dynamics of coherent structures, parts I, II and III. *Quart. Appl. Math.*, XLV:561–590, 1987.

[39] B. W. Spencer, M. W. Heinstein, J. D. Hales, and K. H. Pierson. Multi-length scale algorithms for failure modeling in solid mechanics. Technical Report SAND2008-6499, Sandia National Laboratories, Albuquerque, New Mexico 87185, October 2008.

[40] SIERRA Solid Mechanics Team. Sierra/SolidMechanics 4.26 users guide. Technical report, Computational Solid Mechanics and Structural Dynamics Department, Engineering Sciences Center, Sandia National Laboratories, 2012.

[41] C. Theodoropouls, Y. Qian, and Kevrekidis I. G. Coarse stability and bifurcation analysis using time-steppers: a reaction-diffusion example. *Proc. Natl. Acad. Sci.*, 97(18):9840–9843, 2000.

[42] R. Wang, J. Zhu, Z. Luo, and I. M. Navon. An equation-free, reduced-order modeling approach to tropical pacific simulation. *Advances in Geosciences, Volume 12: Ocean Science (OS)*, 12:1, 2009.